A NEUROCHEMICAL MODEL OF THE BRAIN

A thesis
submitted in fulfilment of
the requirements for the Degree
of
Master of Engineering (Electrical)
in the
University of Canterbury

by
R.B. Dowd, B.E. (Hons)

University of Canterbury

CONTENTS

Page

ABSTRACT

The digital simulation of a network called mew-Brain is described. The network is a neurochemical version of the existing learning system PURR-PUSS.

The organization of computer data for mew-Brain is considered.

Several algorithms which have been developed to enable the incorporation of chemical coding in the network are presented.

Two sets of results are described. These exemplify the electrical and chemical activity of the network.

Some unrelated results are given of a preliminary comparison of ultrasonic imaging devices to determine their suitability in a tactile mobility aid for the blind. Some references encountered during this work are also given.

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION

The learning system PURR-PUSS comprises a number of PUSS predictors (seven at present), each of which receives all or part of a series of occurring events, and PURR, which decides upon an action subsequent to these events on the basis of the predictions made by the individual PUSSes. In doing so, PURR tries to follow a string of successive predictions (an "hypothesis") which leads to a novel event. PURR-PUSS is designed to operate efficiently on a digital computer.

A neurochemical version of PURR-PUSS, called mew-Brain, has recently been described.* Mew-Brain has been designed to follow reasonably closely the neurophysiological structure of the brain, as far as it is known at present, while also remaining functionally similar to PURR-PUSS. Mew-Brain, however, is not designed with digital computing efficiency in mind. Although a parallel version of mew-Brain could be much faster than PURR-PUSS, a (serial) digital simulation is much slower. Mew-Brain has been designed to show that the mechanisms operating in PURR-PUSS, or some similar, could be active in the brain.

---

*Andreae, J.H. 'Mew-Brain, A Neurochemical Version of PURR-PUSS', *In* Man-Machine Studies UC-DSE/9, Nov. 1976, ed. J.H. Andreae. (Elect. Eng. Dept, Univ. of Canty).

Mew-Brain is composed of a network of neurons with interconnections. It possesses both electrical activity in the form of impulses which can travel between neurons via their connections, and chemical activity in the form of chemical codes which are synthesised in certain neurons and can leak along the connections.

In contrast to PURR-PUSS, which uses composite events such as pattern-actions in some of its PUSSes, mew-Brain uses instead both series and parallel events in the contexts of its "cortical-PUSSes". A mew-Brain context comprises two series and two parallel events, a total of three events. Furthermore, hypothesis formation in the form of leak-back of chemical codes is a parallel and hence more efficient process.

Results have been given for a mew-Brain computer simulation written in the Hybrid Operations Interpreter (HOI) interpretive language for the EAI 590 hybrid computer. The purpose of this simulation was to test quickly the proposed mew-Brain mechanisms in a small network. The HOI language was ideal for this, but the simulation was limited to a model with only 45 neurons and both chemical codes and their leak-back paths had to be specified by the programmer.

The lack of algorithms for chemical codes and leak-back paths in the HOI simulation was a serious limitation, so it was decided that a second simulation be attempted. The Assembler language simulation, which is the subject of this thesis, caters for a much larger number of neurons

and contains algorithms prescribing the synthesis of codes and their leak-back paths, based on the structure of the network and its input sequences while in operation.

The very reason that made the HOI language suitable for an initial simulation made it unsuitable for the new simulation. Being interpretive, the language is extremely flexible from the point of view of programming. However, because each line of the program is compiled each time it is executed, HOI runs slowly and requires a large amount of core for its interpreter. With a large mew-Brain network and with the increase in program complexity resulting from the incorporation of chemical coding, data storage would have to be extended to the disc memory and the simulation would become impractically slow.

For this reason the new simulation of mew-Brain is written in Assembler. The program contains algorithms for the synthesis and leak-back of chemical codes, as described here. It is possible to simulate a network of up to 1023 neurons for an unlimited number of firing cycles, although computation is interrupted every 80 firing cycles to print stored results in a graphic display. Such a network, along with 80 firing cycles' worth of output data, occupies almost all of the core storage available in the EAI640's 16k of core memory.

Some results are included here also of a preliminary study carried out to determine the feasibility of developing a tactile mobility aid for the blind. This aid would use ultrasonics to obtain detailed information from a wide sector of the immediate environment of the user.

The information would be presented in a 'plan view' form (p.p.i.) through a matrix of stimulators on the user's abdomen.

The work described here consists of a comparison of various imaging systems which use ultrasonics, considering such parameters as range and bearing resolution, complexity, cost, relative size and signal-to-noise ratio. A discussion is also given of the opposing design philosophies which are encountered in the development of such a device. A list of references which are relevant to this study is included.

Section 1.2 which follows gives an introduction to the neurons and connections of which mew-Brain is composed, and an outline of its mode of operation.

Chapter 2 considers the data organisation in the computer simulation of mew-Brain. Examples are given of the input data required from the operator, to facilitate use of the program.

A detailed description of chemical coding appears in Chapter 3. The four algorithms developed to enable synthesis of codes and chemical leakback appear here.

Results obtained with a sample mew-Brain of 52 neurons and two different input sequences appear in Chapter 4. The results enable the description of several features of the program.

Some possible future developments with mew-Brain are discussed in Chapter 5. Appendix 1 contains program listings, and in Appendix 2 are found some results of the tactile mobility aid study.

1.2 AN INTRODUCTION TO MEW-BRAIN

1.2.1 Neurons in Mew-Brain (refer to Figure 1(a))

1. Input and Output neurons.

These two types of neuron are functionally similar in their firing behaviour. They fire if their firing potentials are above the firing threshold; then, whether they fire or not, their firing potentials are reduced to zero.

Chemical codes called "input SOBs" can be synthesised in input neurons.

2. Context neurons.

These neurons operate in a manner similar to input neurons, except that under certain conditions chemical codes, called context SOBs, are synthesised within them and can leak from them.

3. Integrating neurons.

As well as having a firing potential, integrating neurons have a firing count. If an integrating neuron has a non-zero firing count, it fires whatever its firing potential is and then its firing count is reduced by 1. Apart from the small die-away present in every neuron's firing potential every cycle, its firing potential is unaffected.

If, however, the integrating neuron's firing count is zero, then like non-integrating neurons just described it fires if its firing potential is above threshold and does not if not so.

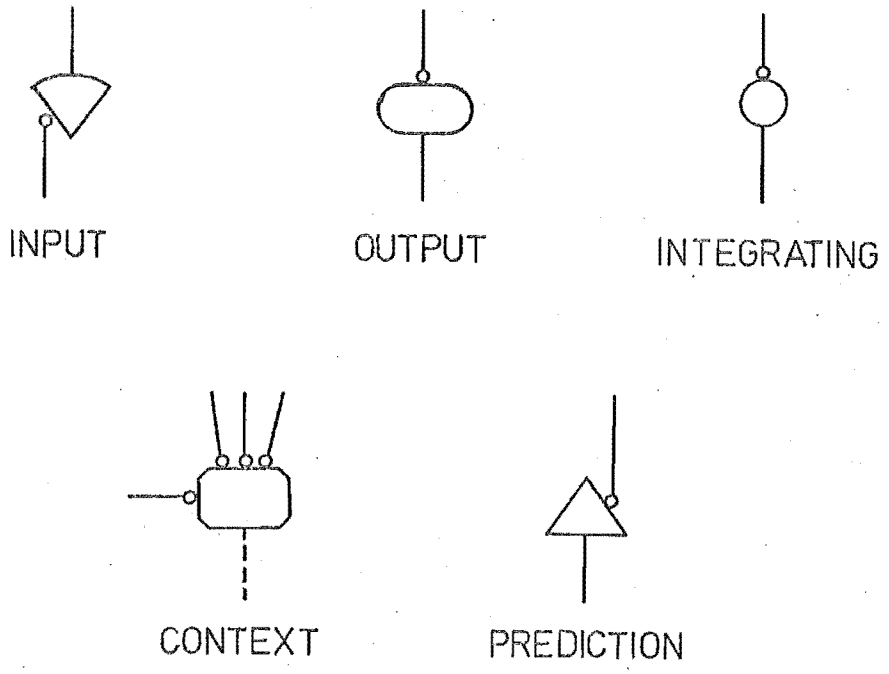


FIGURE 1(a) THE NEURONS IN MEW-BRAIN

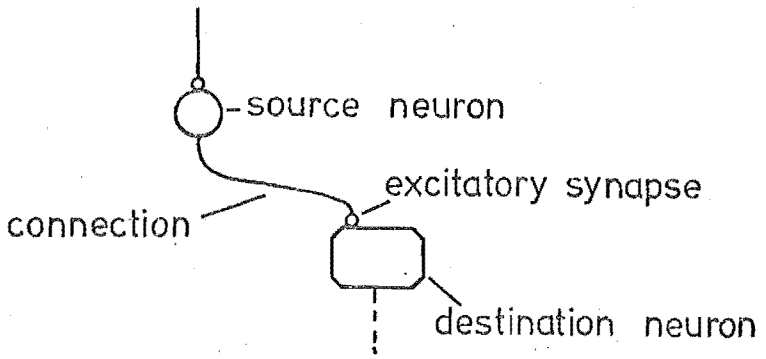


FIGURE 1(b) A CONNECTION IN MEW-BRAIN

FIGURE 1. TERMS USED

Its firing potential is only zeroed if it does fire and in this case its firing count is set to a value called the initial count. An initial count of 9 has been used.

From this the integrating nature of the neuron can be seen. Since the firing potential of any neuron is incremented by the strength of the connection to it from the neuron that has just fired, the strength of connection to an integrating neuron can be chosen so that it will take, say, 9 such increments to bring the integrating neuron's firing potential above its firing threshold, at which time its firing count will be set to 9, ensuring that the neuron will itself fire for 9 cycles. Note also that because at this point the firing potential is zeroed, the integrative action can begin again.

It is such integrating neurons, arranged in three layers, that produce the sequential nature of contexts in mew-Brain.

4. Prediction neurons.

The action of prediction neurons is similar to that of the integrating neurons just described, the only difference being that the initial count is 1.

It is, perhaps, more helpful to view the prediction neuron as being similar to a non-integrating neuron, except that if the neuron does not fire its firing potential is not reduced to zero. So its action is proportional: the higher the strength of a connection to it from a neuron that fires every cycle, the more often will it fire.

1.2.2 Connections in Mew-Brain (refer to Figure 1(b))

Each connection between neurons in mew-Brain has a particular strength of connection associated with it, its value depending on the types of the neurons constituting the source neuron and the destination neuron of the connection.

1.2.3 Chemical Codes in Mew-Brain

The chemical codes called input SOBs that are synthesised in input neurons can leak along the connections in mew-Brain and through all other neurons except context neurons.

Under certain conditions described in Chapter 3, these input SOBs can constitute part of the chemical code synthesised in a context neuron and called a context SOB. Once a context SOB is synthesised, the concentration of this SOB can continually change since it and other context SOB codes can also leak through connections to affect the concentrations of other context SOBs. Since the strength of a connection from a context neuron to a prediction neuron depends on this SOB code concentration, the concentration of this code in turn affects the firing rate of the prediction neuron.

Disregarding for the meantime the requirements under which a context SOB is actually created, Figure 2 shows, as an example, a context neuron with context SOB composed of the input SOBs associated with inputs 1, 2, 3 and 4.

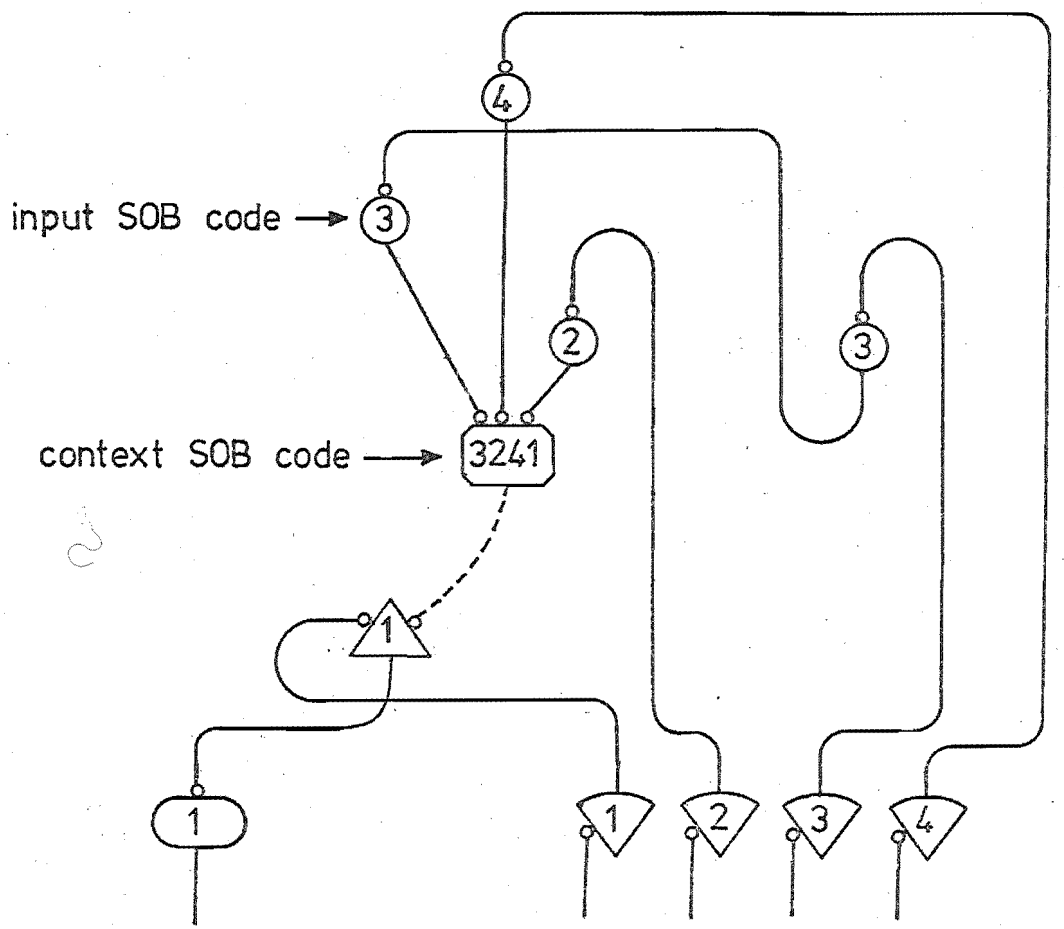


FIGURE 2. CHEMICAL CODES IN MEW-BRAIN

If this context neuron now fires due to the combined effect of increments to its firing potential from all three of its source neurons (note that input 3 will need to arrive before inputs 2 and 4 since it passes through two integrating neurons) then the concentration of its SOB will now affect the firing rate of the connected prediction neuron, which will in turn be transmitted to the output neuron.

The fact that this context neuron has a particular context SOB will mean that the creation of a similar code in an area local to this neuron will be inhibited.

1.2.4 The Operation Cycle of Mew-Brain

The following follows the order of the Assembler program listed in Appendix 1. The flow diagram for the program is shown in Figure 3.

1. Set up network structure.

Data required to specify the neurons and connections for the simulation are given in a manner described in Chapter 2.

2. Find leakage paths for input SOBS.

Now that the network is specified, it is efficient in terms of computer time to describe the paths along which it is possible for input SOBS to leak from each input neuron, rather than to do this every time a particular input sequence arrives. This is done simply by specifying an imagined neuron at every input to mew-Brain. These inputs are given the lowest neuron numbers (every neuron has a number).

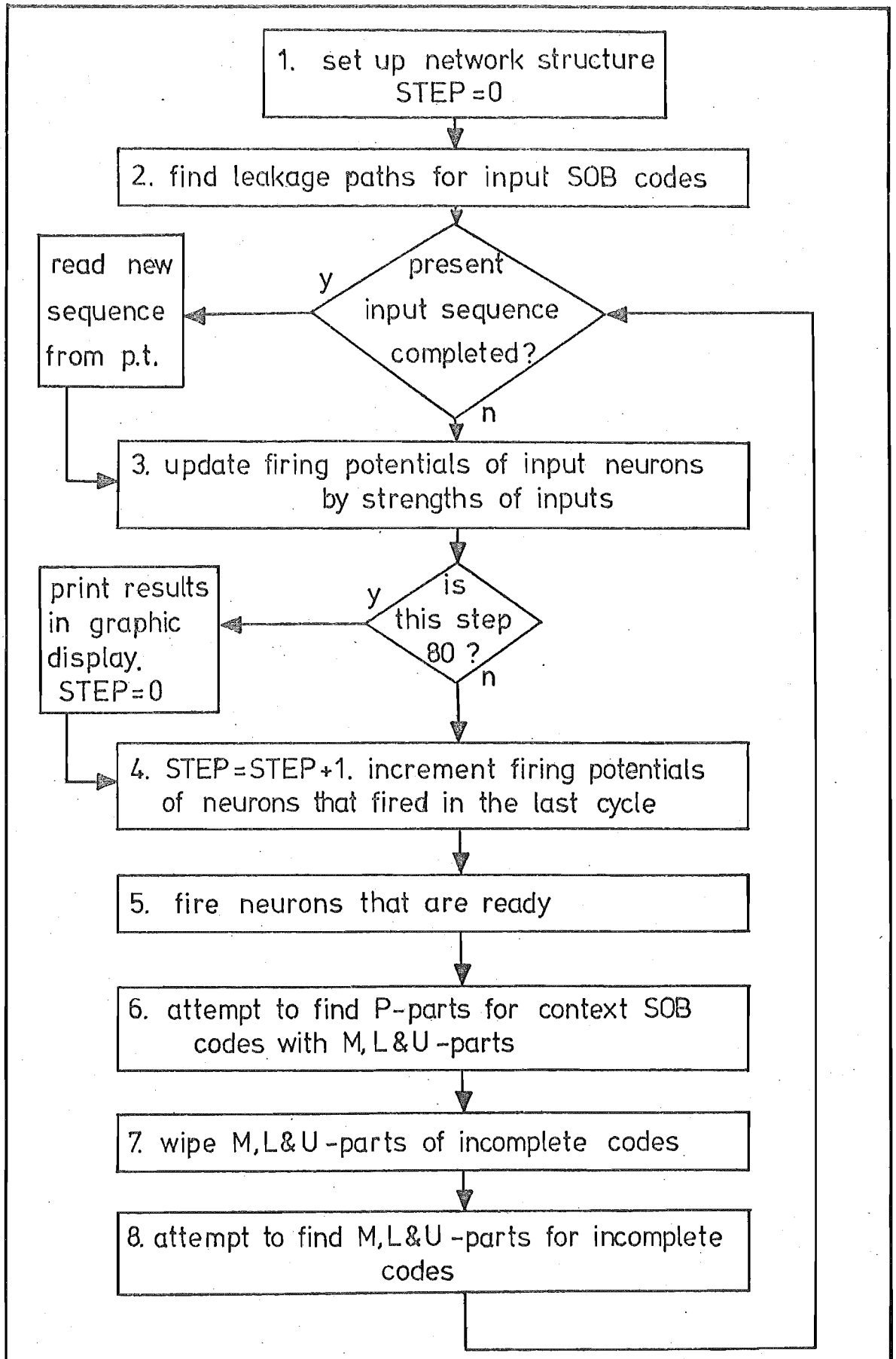


FIGURE 3. FLOW DIAGRAM FOR THE PROGRAM

Now successive connections spreading out from these inputs are scanned and, in each successive destination neuron found (until and excluding a context neuron), is stored the number of the input from which there was a first connection. This number constitutes the input SOB code.

3. Input stage.

This stage is the beginning of an iterative loop. An input sequence is read from paper tape and transmitted to neurons as described in Chapter 2. The tape is read continually while the program computes, except that every 80 steps computation stops to allow data output (described in Chapter 4).

4. Increment stage.

For those neurons that have a flag set to show that they fired in the last operation cycle, the firing potentials of the neurons connected to them are incremented by their respective strengths of connection.

5. Firing stage.

Neurons fire according to the rules in section 1.2.1, and bits are set in a block of core to specify which neurons fired at particular firing cycles. This procedure of data storage and eventual printout is described in Chapter 4.

6. Context SOB code completion.

Two firing cycles are required to create a locally unique context SOB code. During this stage it is determined whether the codes partially synthesised in step 9 of the last cycle can be completed. This is described in section 3.2.1.

7, Wipe uncompleted codes.

Codes that were unable to be completed in step 6 are cleared to allow another attempt at partial synthesis in step 9 of this cycle.

8. Leak-back of completed codes.

During this stage, context SOB codes leak from one to another depending on whether a leak-back exists between particular context neurons. This affects the concentration of the context SOB to which there is a leak-back path. See section 3.2.3.

9. Partial synthesis of new codes.

New codes are partially synthesised if suitable conditions exist. The procedure is described in section 3.2.4.

CHAPTER 2

DATA ORGANIZATION IN MEW-BRAIN

2.1 INPUT DATA REQUIRED

The data required for mew-Brain falls into one of two categories:

(i) Data specifying the structural properties of the network, and

(ii) Data prescribing the input sequences to the structure.

These will be described in turn.

2.1.1 Data Specifying the Structure

The data required in (i) above consists of numerical values, entered for each neuron in turn, for the following variables:

1. Type. The type of the neuron. The allocation of neuron type is

   0 for input or output neurons;
   1 for context neurons;
   2 for integrating neurons;
   3 for prediction neurons.

2. Layer. If the neuron is of type 2 (integrating), its layer in mew-Brain. The layers are

   middle = 1
   lower = 2
   upper = 3

3. Connections. The numbers of the neurons to which there is a connection from the particular neuron, with their respective strengths of connection. The data are given

as a single number: e.g. a connection to neuron 327 with relative strength 24 is stored in a single computer word as the number 32724.

Example 1.

The data specifying the structure shown in Figure 4 would be written as:

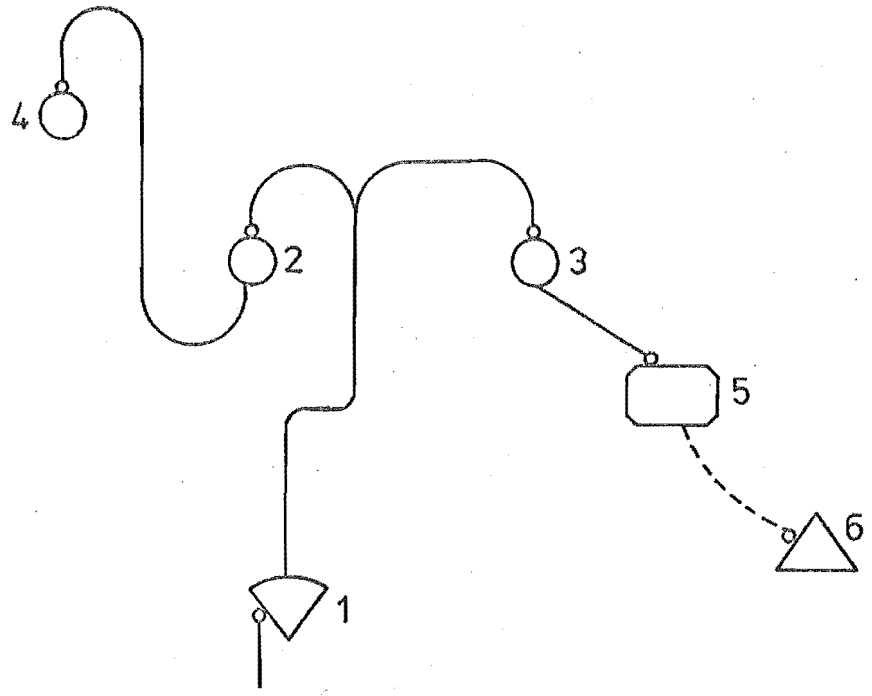
6	The total number of neurons. Sets the size of the blocks described in section 2.2.
Ø,214,314	Neuron 1 is of type Ø (an input neuron) and it has connections to neurons 2 and 3, both with strength 14.
2,2,414	Neuron 2 is integrating (given by the first digit) and hence has a second digit to specify its layer (i.e. lower). It is connected to neuron 4 with strength 14.
2,2,524	Neuron 3.
2,1,Ø	Neuron 4 is an integrating neuron in the middle layer without a destination neuron.
1,6ØØ	Neuron 5 is a context neuron having a potential connection (i.e. a connection with zero strength) to neuron 6.
3,Ø	Neuron 6.

Note: All data is octal and integer.

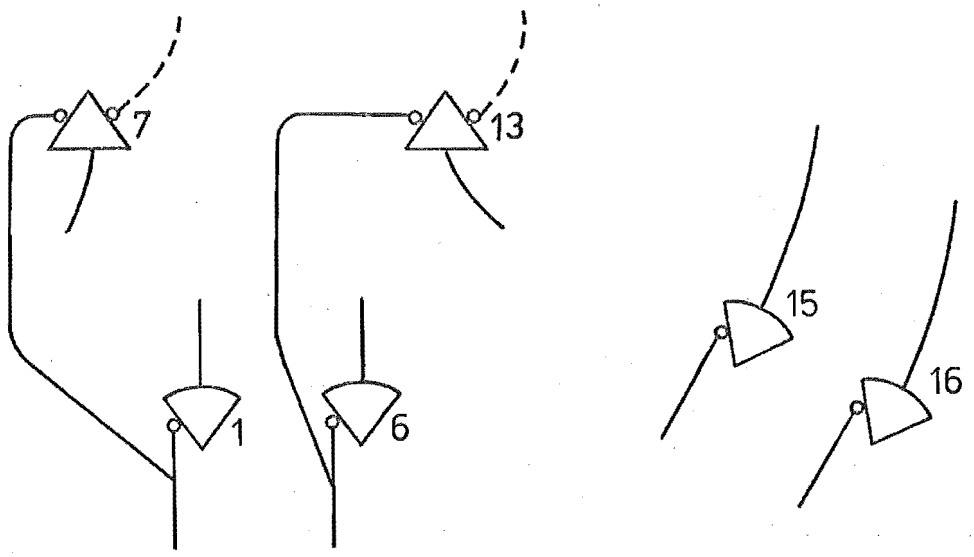
Data for each neuron is separated by means of a carriage return/line feed.

2.1.2 Data Specifying Input Sequences

An input pulse is in effect an increment to the firing potential of the neuron to which an input is connected. The data required consists of:



EXAMPLE 1.



EXAMPLE 2.

FIGURE 4. EXAMPLES OF DATA INPUT

1. The neuron number specifying the neuron to which an input pulse is being applied.
2. The magnitude of the increment to its firing potential. This can be considered as the strength of the connection from an input to an input neuron.
3. The number of such identical input pulses in a sequence.

The data specified in 1 and 2 are of a similar form to that required to specify the network structure, as can be seen from example 2.

Example 2.

The structure is shown in Figure 4.

11,175,775,1575	11_8 (decimal 9) pulses of input are specified in this sequence, with the firing potentials of neurons 1, 7 and 15 being incremented by 75 each time.
11,675,1375,1675	Then, similarly for neurons 6, 13 and 16.
2Ø,Ø	Following this there are 20 firing cycles during which no input is applied.

Note: Since the firing threshold is set at 72, an increment of 75 to a neuron's firing potential ensures that it will fire during the next firing cycle.

2.2 DATA STORAGE

Mew-Brain data are stored in a series of blocks in core, following on directly from the end of the program. The size of the blocks depends on the number of neurons in the simulation, specified by the input data. Figure 5 shows the layout of these blocks.

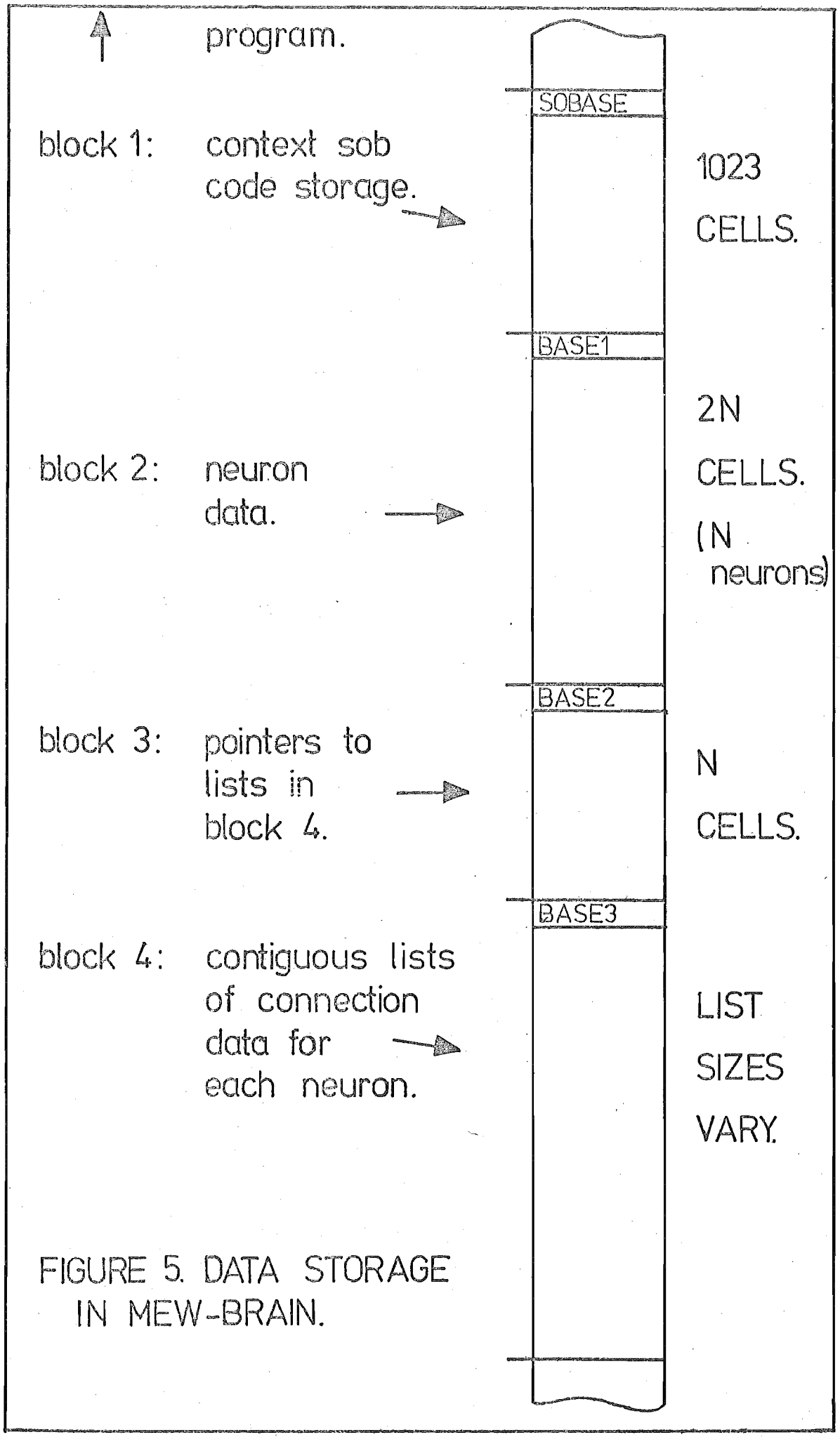


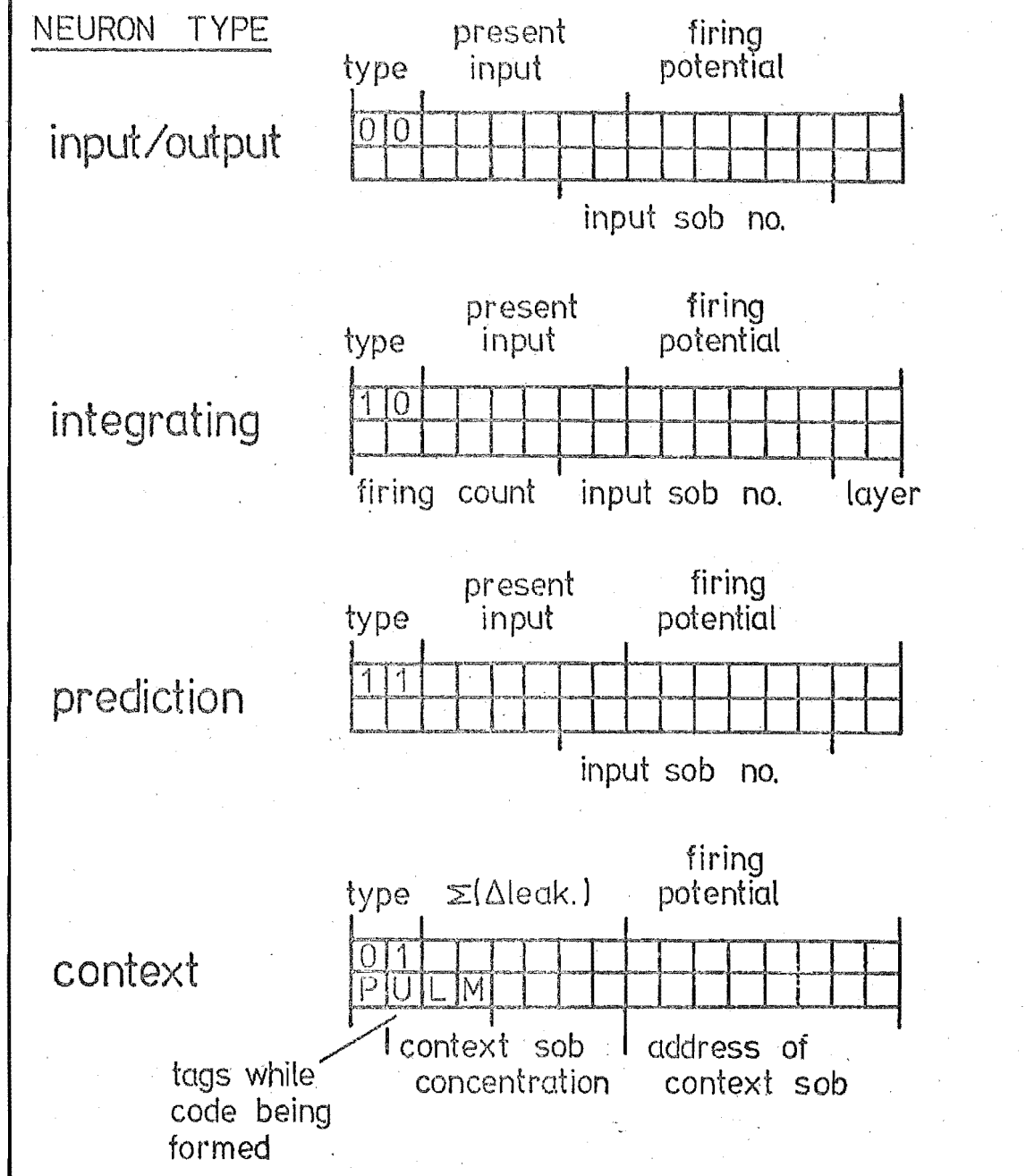
FIGURE 5. DATA STORAGE IN MEW-BRAIN.

Block 1 is used for storing chemical codes and will be described later in Chapter 3.

Block 2 is addressed so that, for each neuron present, two consecutive computer words are available. Addressing is performed indirectly, using the location 'BASE1'. The organization of data within these two words differs slightly depending on the type of the neuron, and is shown in detail in Figure 6(a). The references to input and context SOB codes in the Figure pertain to chemical coding and again will be described in Chapter 3.

The firing potential, present in all neurons, is allocated eight binary digits to give a maximum nagnitude of 377_8. the last two octal digits can be considered fractional, so that the firing threshold, for example, is $\emptyset 72_8$, and the die-away of firing potential is $\emptyset\emptyset 6_8$ for each cycle.

The six bits allocated for 'present input sequence' in all but context neurons (though at present only used in input and prediction neurons) enable an increment to the firing potential of the neuron in which they are stored to be applied during the input stage without requiring new input data to be read from the paper tape every cycle. This is useful if a particular input sequence lasts for more than one firing cycle (as is usually the case). The six bits enable a maximum increment to the firing potential of 77_8, enough to guarantee that the firing potential of a neuron will be above the firing threshold once an input is applied. The six bits are cleared when the sequence ends.



(a) DATA STORED IN BLOCK 2.

(b) INFORMATION WORD IN LISTS OF BLOCK 4.

FIGURE 6. DATA STORAGE IN DETAIL.

As described in the introduction, once an integrating neuron begins to fire, it does so for the number of pulses specified by the initial count (the same for all integrating neurons in the three layers). Its firing count decreases by 1 each cycle and it fires until its firing count is zero. The six bits allocated in the second word for each integrating neuron in Figure 6(a) enable a maximum initial count of 63, although at present the initial count is set at 9.

Note that because a prediction neuron can be considered to be similar to a non-integrating neuron, except that its firing potential is not zeroed if it does not fire, it is not necessary to store its firing count which is always either 1 or 0.

As described in section 1.2.2, each neuron has one or more neurons to which it is connected, each connection having a strength dependent on the types of the neurons at either end. The actual strengths of the connections are as octal integers:

input to input neuron	= 75
input to prediction neuron	= 75
input neuron to context neuron	= 06
input neuron to integrating neuron	= 14
integrating neuron to integrating neuron	= 14
integrating neuron to context neuron	= 24
context neuron to prediction neuron	= 0
prediction neuron to output neuron	= 75
output neuron to input neuron	= 75

Because the number of neurons having connections from a particular neuron varies, list processing is used to store such data rather than setting aside similar sized blocks of core. For each neuron, a pointer gives the address of the beginning of its list of connection and strength of

connection data. Each information word on this list is a composite numeral made up of the number of a neuron to which there is a connection and the strength of the connection, as shown in Example 1; the digital word has the form shown in Figure 6(b).

Next to each such information word is stored another word giving the address of the next information word on the same list. Hence a list is only as long as is required to store the connection information relevant to each neuron. The standard list processing subroutines PUSH and SCAN of the TRAC language implementation* are used to set up the lists initially and to search through connections, respectively. The lists follow on, one from another in Block 4 of Figure 5 while the 15-bit pointers, being the addresses of the first word on each neuron's list, reside in Block 3. The pointers can be indirectly addressed from the cell 'BASE2'.

When mew-Brain is in operation, the fact that a particular neuron has fired is required to be stored for one firing cycle. The single free bit in the pointer word of Block 3 for this particular neuron is used as a flag for this purpose.

---

*Mooers, C.N. 'TRAC, a procedure-describing language for the reactive-type writer', Comm. ACM, 1965, p.215.

CHAPTER 3

CHEMICAL CODING IN MEW-BRAIN

3.1 DESCRIPTION OF CHEMICAL CODES

A context neuron SOB code comprises four input SOB codes, which have leaked from four specific groups of neurons. One input SOB, leaked from an upper layer integrating neuron, constitutes the U part of a context code; that from a middle layer integrating neuron, the M part; from a lower layer integrating neuron the L part; and from a prediction neuron the P part. The completed context SOB code is designated SOB-MLUP and is illustrated in Figure 7(a).

This SOB-MLUP is created if (i) there is already no code created and (ii) the context has been fired, i.e. the upper, middle and lower layer integrating neurons have fired simultaneously so that the firing potential of the context neuron to which they are connected has been incremented sufficiently for it to fire in the next firing cycle, and (iii) as well as the context neuron firing in the next firing cycle, a prediction neuron to which the context neuron has a potential connection also fires.

Once the SOB-MLUP is created:

(i) any different context that fires into this context neuron has no effect on this particular code.

(There may be more than just one connection to the context neuron from each layer of integrating neurons, and perhaps more than one prediction neuron to which there is a potential connection.)

(ii) The connection between the context neuron and the prediction neuron immediately becomes established, since the code now has a concentration set at an initial value (and, as described in section 1.2.3, this affects the strength of connection).

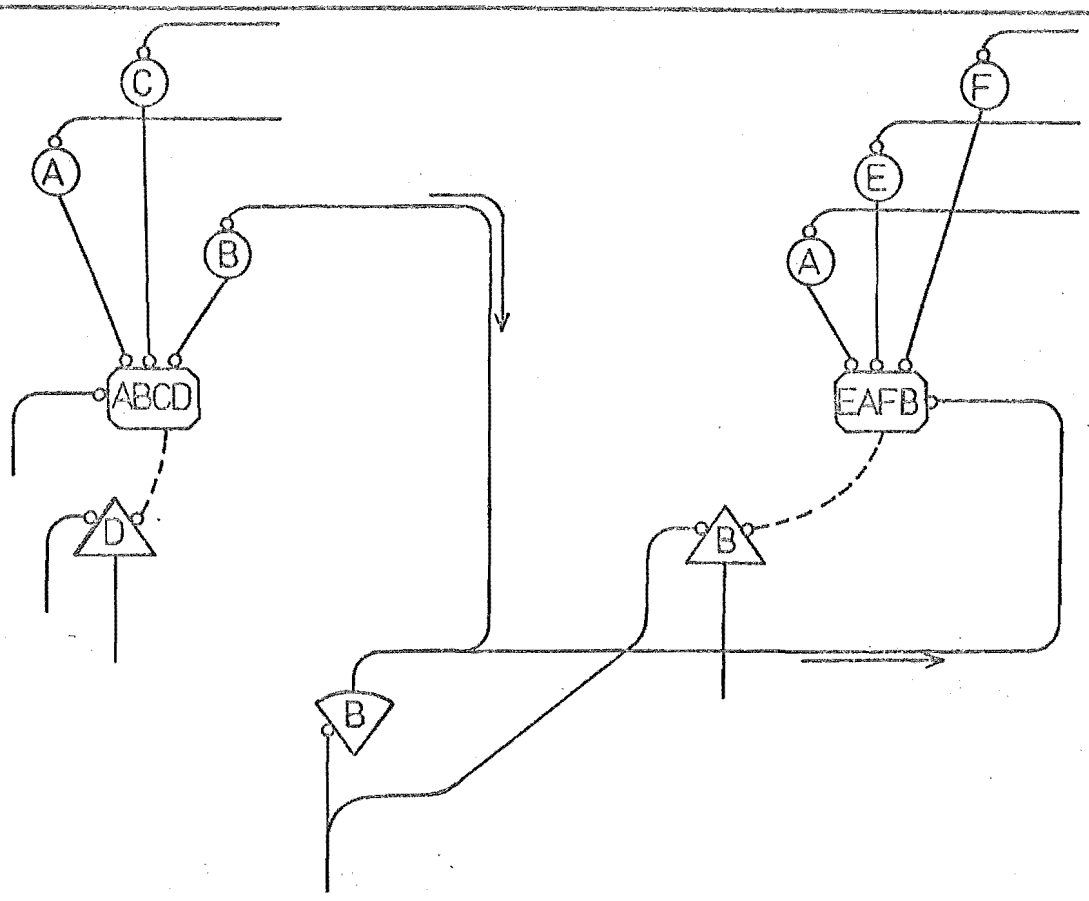
(iii) An inhibitory version of SOB-MLUP is created and this SOB-MLUP' stops the same SOB-MLUP from being created in context neurons having potential connections to the same prediction neuron, by leaking to them through the prediction neuron.

(iv) The code can leak through integrating neurons back to input neurons and so out through other integrating neurons to other context neurons, causing the chemical codes in these context neurons to be synthesised in greater concentration.

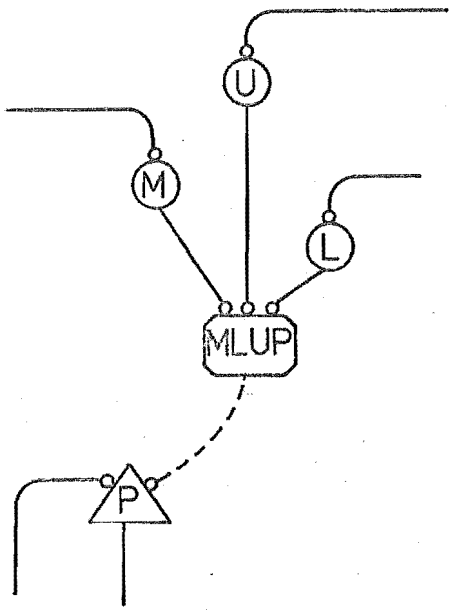
Figure 7(b) shows the possible path by which the process referred to in (iv) could occur for the SOB-ABCD.

It can be seen that leak-back will occur for those context neurons with SOB codes having a match between the L part of one and the P part of another; and a match between the M part of the same one and the L part of the other. That is, in the example of Figure 7(b) the code SOB-ABCD leaks to the context neuron having code SOB-EAFB.

The code SOB-ABCD leaks no further than to the context neuron having code SOB-EAFB, but SOB-EAFB may leak-back further to other context neurons having matching SOB codes.



(b) A LEAKBACK PATH



(a) CONTEXT SOB CODE MLUP

FIGURE 7. DESCRIPTION OF CODING

Successive leak-back can continue until SOB code concentrations have died away to an ambient level, which they then maintain. (At present the die away of SOB concentrations is set at about 1% per firing cycle, calculated probabilistically.)

The detailed algorithms for the synthesis and leak-back of chemical codes now follow. The order in which these take place, in relation to the rest of the firing cycle, is given in section 1.2.4.

3.2 CODE SYNTHESIS AND LEAK-BACK ALGORITHMS

Block 1 of Figure 5 is shown enlarged in Figure 8. The block is broken up into 4 sub-blocks, one for each of the possible parts of a context SOB code, M, L, U and P. The reasons for such a storage configuration are as follows:

(i) The program can cater for a maximum of 1023 neurons. Since it is certain, for any practical simulation, that less than one quarter of these neurons will be input neurons in which input SOB codes are synthesised, then no input SOB code will be a figure greater than 256 or 377_8. Such a figure requires 8 binary digits for storage and, in fact, it can be seen from Figure 6(a) that eight bits are always allocated for recording input SOBs in the neurons that such SOBs can leak through. It is not certain, however, that less than one eighth of the neurons will be input neurons, so no smaller number of bits can safely be used.

It is obvious then that it is not possible to store four such input SOB codes, in the form of a context SOB code, in one 16-bit computer word.

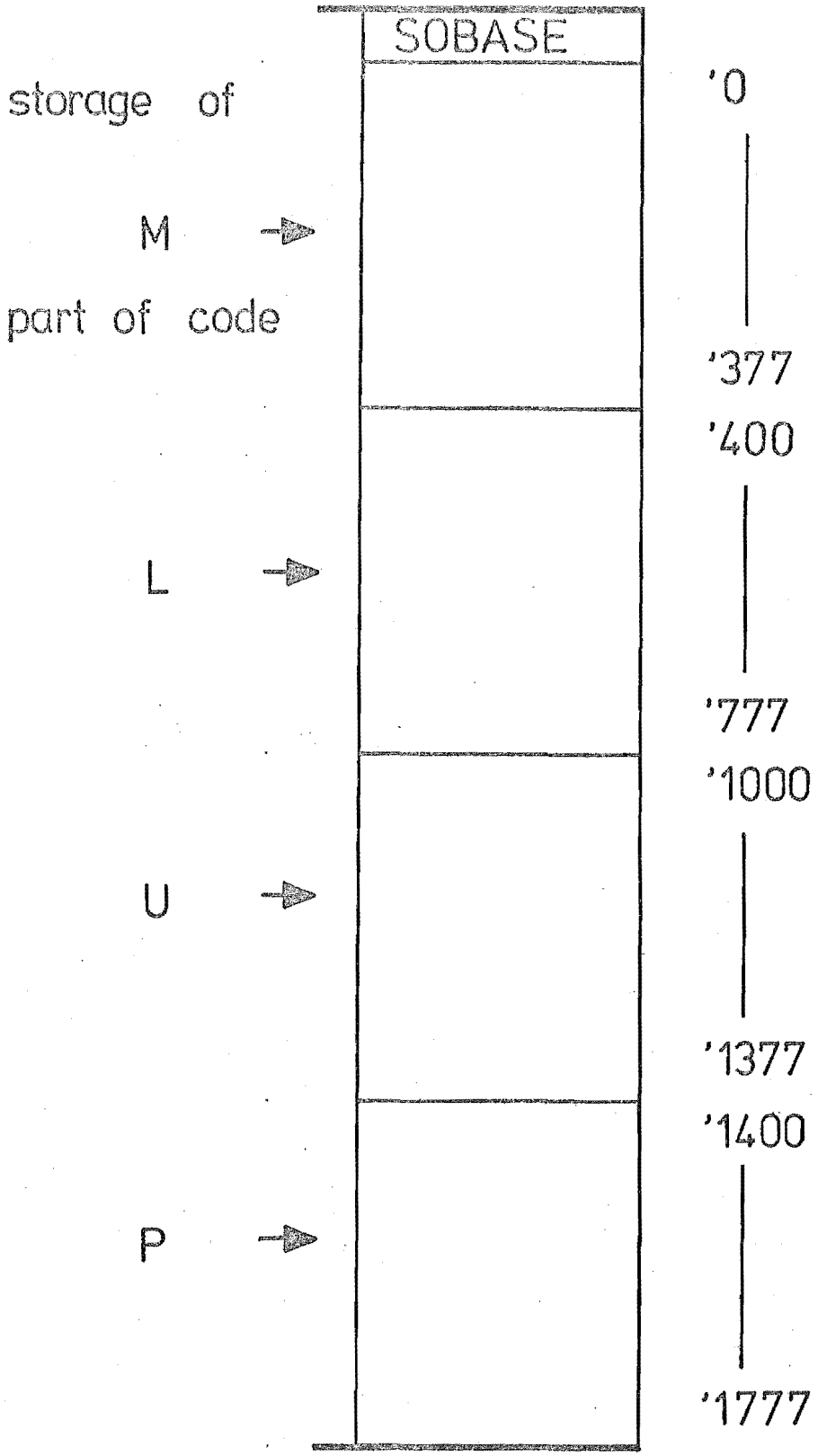


FIGURE 8. BLOCK 1: STORAGE OF SOB-CODES.

Although it would be possible to store half the context SOB in a 16-bit computer word, the advantages further to be mentioned, of using one computer word for one quarter of the code, outweigh any saving in storage obtained.

(ii) Each part of the SOB code for a particular context neuron is stored exactly 400_8 cells from the last and all parts of the code can be addressed quickly relative to the cell SOBASE. This means that only the address of the first part of a context SOB code (the M part) need be stored in the context neuron associated with it. This address will itself be a number less than 400_8 and hence requires a maximum of 8 bits for storage (see Figure 6(a)). To obtain the L part of the code for the context neuron, 400_8 is added to the address. The same procedure applied again gives the U and P parts.

(iii) Note that by successively adding 400_8 to an address between zero and 377_8, the digits making up the basic address, i.e. the first eight binary digits, remain unchanged. This is useful for constructing the address of codes and will be described later.

(iv) Again, since the ratio of context neurons to the total number of neurons will be between one quarter and one eighth, using eight bits to give the maximum number of context SOB codes possible as 400_8 is not unreasonable in terms of the storage space allocated.

In the algorithms that follow, the contents of a cell (i.e. a 16-bit computer word) with octal address A, is specified as C(A). Parts of a particular cell are denoted

as fields, using the symbolism Ba-b. This specifies the field as the bits of the cell numbered from a to b inclusive, where Ø is the most significant bit (MSB) and 15 is the least significant bit (LSB).

Since data for a neuron n is stored in a cell in one of the blocks of Figure 5, which is a multiple of n places from a base cell (e.g. SOBASE, BASE1), that particular data can be accessed using index register addressing indirectly from the base cell. This results in the contents of the index register (n or a multiple of n) being added to the address of the base cell to give the address of the cell where the required data for n is stored.

In the following algorithms, such indirect addressing occurs wherever cells are specified in the form C(BASE+n).

The program for these algorithms appears in Appendix 1. Comment labelling in the program follows the labelling used in the algorithms.

3.2.1 Algorithm A: Completion of Chemical Codes

Algorithm A follows directly after the firing stage. Two firing cycles are required to synthesise a code completely, as described earlier. In this algorithm, those context neurons for which a context "fired" in this firing cycle (i.e. integrating neurons in all three layers fired into it in the cycle before) are examined to determine whether a P-part for the code can be obtained. Hence, possible M, L and U parts have already been found (in "Synthesis of New Codes" stage of the last cycle) and such context neurons already have M, L and U tags set.

N = total number of neurons.

A. For neuron n from 1 to N, do the following:-
A.1 (Check n just fired) Continue if BØ of C(BASE2+n)=1.
A.2 (Check n is context neuron) Let x=BASE1+2n.
Continue if BØ-1 of C(x-1) = Ø1.
A.3 (Check n has M,L,U tags)
Continue if B1-3 of C(x) = 111.
A.4 (Scan connections of n) For each information word C(e) on the list whose pointer is B1-15 of C(BASE2+n), do the following:-
A.4.1 (n connected to prediction neuron?) Let p=BØ-9 of C(e) and r=BASE1+2p. Continue if BØ-1 of C(r-1) = 11.
A.4.2 (p just fired?) Continue if BØ of C(BASE2+p) = 1.
A.4.3 (Store P-part of context SOB)
C(SOBASE+14ØØ+B8-15 of C(x)) ← B6-13 of C(r).
A.4.4 (Check local uniqueness) For neuron m from 1 to N, except n, do the following:-
A.4.4.1 (Check m context neuron) Let y = BASE1+2m.
Continue if BØ-1 of C(y-1) = Ø1.
A.4.4.2 (Check m has P-tag) Continue if BØ of C(y) = 1.
A.4.4.3 (Search for connection to p) C(f) is an information word on the list whose pointer is B1-15 of C(BASE2+m). For each C(f), q = BØ-9 of C(f). Continue only if at least one q = p.
A.4.4.4 (Compare codes of n and m) Continue if C(SOBASE+z+B8-15 of C(y)) ≠ C(SOBASE+z+B8-15 of C(x)) for any one of z = Ø,4ØØ,1ØØØ,14ØØ.
A.4.5 (Completed Code. Set P-tag and initial SOB concentration) BØ-7 of C(x) ← 11ØØØØØØ.

3.2.2 Algorithm B: Wipe Tags

If, for a particular neuron, a chemical code failed to be completed in algorithm A, then any of the tags for M, L or U, set during the last firing cycle, must be wiped to allow a new attempt at code synthesis this cycle, in algorithm D.

B. For neuron n from 1 to N, do the following:-
B.1 (Check n is context neuron) Let x = BASE1+2n.
Continue if BØ-1 of C(x-1) = Ø1.
B.2 (Check no P-tag) Continue if BØ of C(x) = Ø.
B.3 (Wipe tags) B1-3 of C(x) ← ØØØ.

3.2.3 Algorithm C: Leak-Back

This algorithm finds leak-back paths and the amount of the leak-back for each path found. It also adjusts the

strength of the connection from a context neuron to its associated prediction neuron, depending on the concentration of the SOB code in the context neuron.

Probabilistic routines have been used in this algorithm to determine the amount of the leak-back between particular codes, and the die-away of these codes. For the leak-back case, a constant increment (Δleakback) is added with decreasing probability (rather than a decreasing increment being added continually). This procedure stops any loss of significant digits. For the die-away of codes, an increment Δdie-away is subtracted in a similar manner. The constants Δleakback and Δdie-away affect the amount of leakback and the speed of die-away, respectively, and can be varied.

The total increment to a SOB code concentration due to leak-back for one firing cycle ($\Sigma\Delta$leakback) is stored in a list until every SOB code has been considered. All such increments are then added to their respective SOB code concentrations, and the parallel nature of the process is maintained. The entry on the increment 'list' for neuron n is actually

B2-7 of C(BASE1 + 2n-1).

C.1 (Find leak-back paths and amount of leak-back)
For neuron n from 1 to N, do the following:-
C.1.1 (Check n is context neuron) Let x = BASE1+2n.
Continue if BØ-1 of C(x-1) = Ø1.
C.1.2 (Check for completed code) Continue if
BØ of C(x) = 1.
C.1.3 For neuron m from 1 to N, except n, do the
following:-
C.1.3.1 (Check m is context neuron) Let y = BASE1+2m.
Continue if BØ-1 of C(y-1) = Ø1.
C.1.3.2 (Check for completed code) Continue if
BØ of C(y) = 1.

C.1.3.3 (Is there an L-P match?) Continue if
C(SOBASE+14ØØ+B8-15 of C(y))
= C(SOBASE+4ØØ+B8-15 of C(x)).

C.1.3.4 (Is there an M-L match?) Continue if
C(SOBASE+4ØØ+B8-15 of C(y))
= C(SOBASE+B8-15 of C(x)).

C.1.3.5 (Determine Δconc) Continue if Δconc > Ø, where
Δconc = (B1-7 of C(x)) - (B1-7 of C(y)).

C.1.3.6 (Probabilistic leak-back)
ΣΔleak-back ← Ø.
(C(R) is computed random number of same range as Δconc) Do the following for 5 numbers C(R):
If C(R) < Δconc, ΣΔleakback ← ΣΔleakback + Δleakback and Δconc ← Δconc - Δleakback,
except when Δconc < Δleakback, in which case ΣΔleakback ← ΣΔleakback + Δconc, and exit to C.1.3.7.

C.1.3.7 (Add to present ΣΔleakback)
B2-7 of C(y-1) ← B2-7 of C(y-1) + ΣΔleakback.

C.2 (Update SOB concentrations and strengths of connections to prediction neurons) For neuron n from 1 to N, do the following:-

C.2.1 (Check n is context neuron) Let z = BASE1+2n.
Continue if BØ-1 of C(z-1) = Ø1.

C.2.2 (Check for completed code) Continue if BØ of C(z) = 1.

C.2.3 (Update SOB concentration)
B1-7 of C(z) ← B1-7 of C(z) + B2-7 of C(z-1).
(Die-away) If a C(R) < B1-7 of C(z),
B1-7 of C(z) ← B1-7 of C(z) - Δdie-away.
B2-7 of C(z-1) ← Ø.

C.2.4 (Update strength of connection)
For each information word C(e) on the list whose pointer is B1-15 of C(BASE2+n) do the following:-

C.2.4.1 (Check n connected to prediction neuron)
Let p = BØ-9 of C(e),
r = BASE1 + 2p-1.
Continue if BØ-1 of C(r) = 11.

C.2.4.2 (Check magnitude of SOB concentration)
If B1-7 of C(z) ≥ $1ØØ_8$, B1Ø-15 of C(e) ← 77_8.

C.2.4.3 (Copy SOB concentration into strength of connection)
If B1-7 of C(z) < $1ØØ_8$,
then B1Ø-15 of C(e) ← B1-7 of C(e).

3.2.4 Algorithm D: Partial Synthesis of New Codes

In this algorithm, M, L and U parts of a possible new context SOB code are found. Since only one input SOB stored in an integrating neuron from each layer can constitute part of a particular context SOB code, if there is more than one integrating from a particular layer connected to the context neuron, a random choice is made

between the integrating neurons.

D. (Initialize) SOBNUM ← 1.
(Cycle) For neuron n from 1 to N, do the following:-
D.1 (Check n just fired) Continue if BØ of C(BASE2+n) = 1.
D.2 (Check n is integrating neuron) Let x = BASE1 + 2n.
Continue if BØ-1 of C(x-1) = 1Ø
D.3 (Scan connections of n) For each information word C(e) on the list whose pointer is B1-15 of C(BASE2+n), do the following:-
D.3.1 (Is n connected to context neuron?) Let p = BØ-9 of C(e) and r = BASE1+2p. Continue if BØ-1 of C(r-1) = Ø1.
D.3.2 (Check code incomplete) Continue if BØ of C(r) = Ø.
D.3.3 (Is part of context code associated with layer of n already found?)
t ← B14-15 of C(x). Now, if t = 11, t ← 1ØØ.
(Check tags held already)
If (B1-3 of C(r))AND(t) ≠ Ø, continue only if B15 of a computed random number = Ø.
D.3.4 (Set tag and check that neuron has code address)
B1-3 of C(r) ← (B1-3 of C(r))OR(t).
If B8-15 of C(r) = Ø, B8-15 of C(r) ← SOBNUM and SOBNUM ← SOBNUM + 1.
D.3.5 (Store code) z ← t with LSB truncated.
C(SOBASE+(z X 4ØØ) + B8-15 of C(r)) ← B6-13 of C(x).

CHAPTER 4

RESULTS

4.1 FORMAT AND STORAGE OF RESULTS

The format of mew-Brain results is a matrix-type display with graduations for the number of firing cycles on the horizontal axis and neuron numbers on the vertical axis. A dot is printed at the appropriate coordinate if a particular neuron fired during a particular firing cycle. Illustrative print-outs appear in Figures 12 and 14.

The display is printed directly at the computer typewriter by the Assembler program. However, the results are not printed as mew-Brain computes but are stored in an area of core memory as they are obtained. At intervals of 80 firing cycles, computation is interrupted and a display like that of Figures 12 and 14 printed. Computation then continues.

The reason for this arrangement of data output is that the mew-Brain program iterates through all neurons for each firing cycle, rather than vice-versa (i.e. we are simulating a group of neurons working in parallel). Therefore, to print results after each firing cycle either the typewriter carriage would have to be able to move vertically up and down the page, or neuron numbers would need to be placed on the horizontal axis. The computer typewriter does not allow the first, and the second would

limit the number of neurons to a number far below the 1023 allowed by the program.

Five 16-bit computer words are allocated in core memory for each neuron. If the neuron fires during a particular firing cycle, then the appropriate one of the 80 bits is set. In this way, the print-out routine is informed at the end of each 80 cycle series which neurons fired and when. With capacity of 1023 neurons, these 5 words per neuron take 5k of core. Adding to this 10·5k for program and data when a 1023 neuron simulation is running, the total is just less than the available 16k of memory on the EAI640 computer in the Department. Another reason for choosing 80 cycles is that 86 characters fit across the standard DECwriter (computer typewriter) page, which leaves 6 characters for the neuron number when one space is allowed for each firing cycle.

4.2 EXAMPLES OF RESULTS

Figure 9 shows a small mew-Brain, for which results have been obtained using the Assembler program. The network structure is similar to that run with the earlier HOI program except that several connections have been added to illustrate special features of the new program. Note that neuron 54 is connected to six different integrating neurons as source neurons and that neuron 60 has four. Also, there is a potential connection from context neuron 54 to prediction neuron 12.

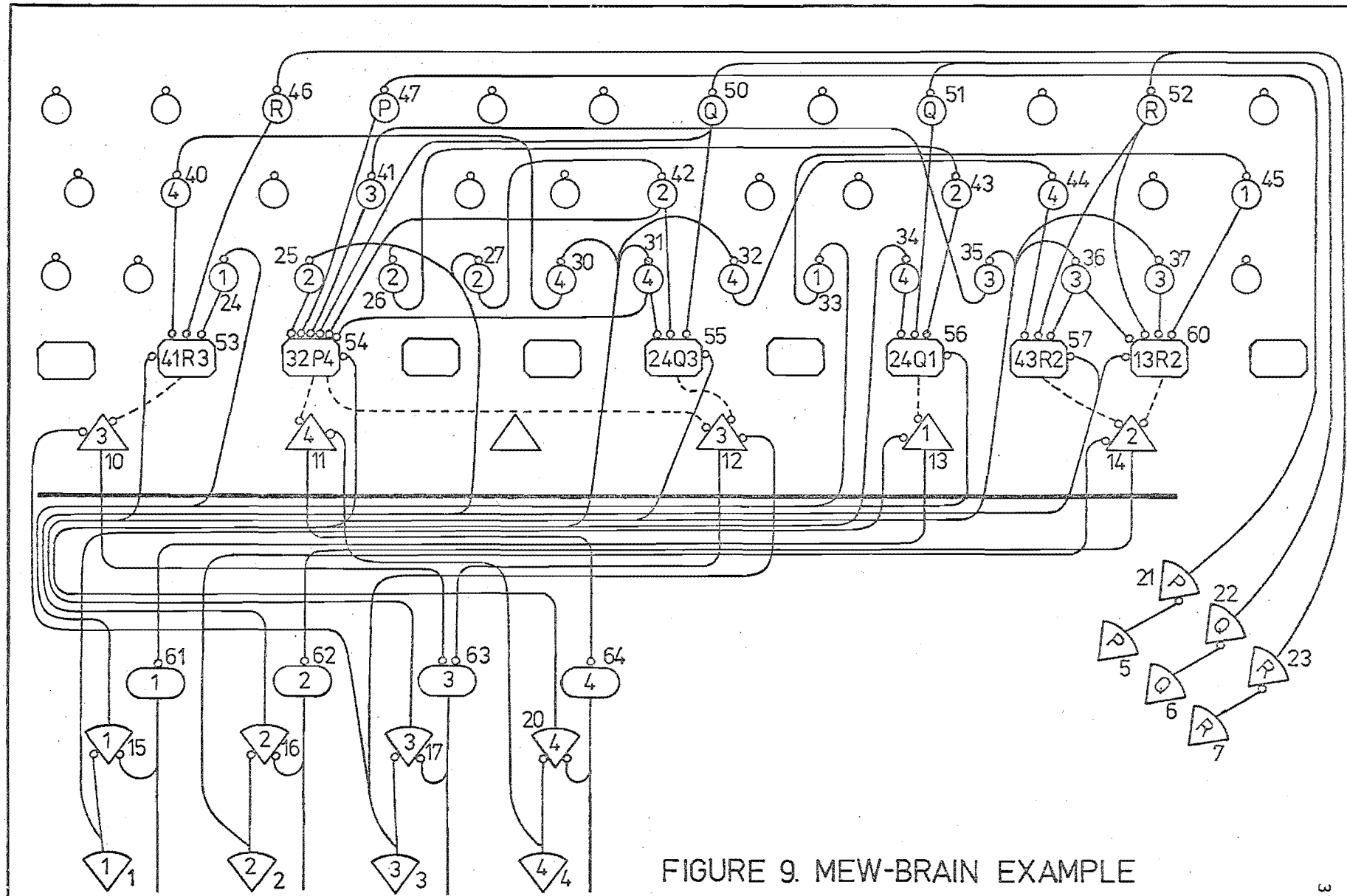


FIGURE 9. MEW-BRAIN EXAMPLE

The numbers written within neurons in Figure 9 are the SOB codes associated with these neurons as a consequence of the action of the network. Those connections which are initially potential connections (i.e. they have zero strength) are shown as broken lines. The data which describe the structure of this network are shown in Figure 10.

In Figure 12 is shown the electrical activity of this network for input data sequence A (shown in Figure 11) over 130 firing cycles. The vertical axis shows neuron numbers in octal. Note that inputs are specified as input neurons and have the numbers 1 - 7. The firing pattern to the right of these numbers is the input sequence. Neurons numbered 15 - 23 are the input neurons. Note that they each fire one cycle later than the input from which they have a connection.

Neurons 24 - 52 are integrating neurons with a firing count set at 9. Note how neuron 25, for example, requires 9 pulses from input neuron 15 before it will fire, and then it fires for the number of its firing count. The action of the layers of integrating neurons can be seen; a sequence that passes through two integrating neurons becomes delayed twice compared to an input sequence, while one that passes through one integrating neuron is delayed by a 9 pulse interval only once. In this way, input sequences arrive at context neurons in a pattern corresponding to the context of two parallel events preceded by one event.

```
001: 64
002: 0,1575,1375
003: 0,1675,1475
004: 0,1775,1275,1075
005: 0,2075,1175
006: 0,2175
007: 0,2275
008: 0,2375
009: 3,6375
010: 3,6475
011: 3,6375
012: 3,6175
013: 3,6275
014: 0,2414,3314,5606
015: 0,2514,2614,2714,5706,6006
016: 0,5306,5506,3514,3614,3714
017: 0,5406,3014,3114,3214,3414
018: 0,4714
019: 0,5014,5114
020: 0,4614,5214
021: 2,2,5324
022: 2,2,5424
023: 2,2,4314
024: 2,2,4214
025: 2,2,4014
026: 2,2,5524,5424
027: 2,2,4414
028: 2,2,4514
029: 2,2,5624
030: 2,2,4114
031: 2,2,5724,6024
032: 2,2,6024
033: 2,1,5324
034: 2,1,5424
035: 2,1,5524,5424
036: 2,1,5624
037: 2,1,5724
038: 2,1,6024
039: 2,3,5324
040: 2,3,5424
041: 2,3,5524,5424
042: 2,3,5624
043: 2,3,5724,6024
044: 1,1000
045: 1,1100,1200
046: 1,1200
047: 1,1300
048: 1,1400
049: 1,1400
050: 0,1575
051: 0,1675
052: 0,1775
053: 0,2075
```

FIGURE 10. DATA FOR MEW-BRAIN EXAMPLE

```
001: 11,475,675
002: 11,375,775
003: 11,275,575
004: 11,475,675
005: 11,375,775
006: 11,0
007: 11,475,675
008: 11,375,775
009: 11,275,575
010: 11,475,675
011: 11,0
012: 11,275,575
013: 11,475,675
014: 100,0
```

Data sequence B

```
001: 11,275,575
002: 11,475,675
003: 11,375,775
004: 11,275,575
005: 11,475,675
006: 11,375,775
007: 11,275,575
008: 11,475,675
009: 11,175,775
010: 11,375,775
011: 11,275,575
012: 11,475,676
013: 100,0
```

Data sequence A

FIGURE 11. INPUT SEQUENCE DATA

4.2.1 Synthesis of a Chemical Code

The first time that conditions exist for a chemical code to be partially synthesised is at t = 20 (Figure 12). Here neurons 31, 42 and 50 fire for the first time and algorithm D is able to provide M, L and U parts for the context SOB code that (hopefully) will reside in neuron 55. These M, L and U parts will actually be the input SOBs stored in the three integrating neurons 31, 42 and 50, which have already leaked there from their respective inputs.

In the next firing cycle, t = 21, conditions exist for algorithm A to complete this context SOB code. This is possible because neuron 12, the prediction neuron with potential connection from context 55, is being fired by input 3. And at the same time neuron 55 fires (this is expected since we have already seen that at least three neurons fired into it the cycle before, so its firing potential will be above threshold). Therefore the code is completed and is given an initial concentration of $1\emptyset\emptyset_8$, and the potential connection to prediction neuron 12 is updated to this value also. Notice that in Figure 13 which shows the context SOB code concentrations, code 24Q3 is set to its initial value at t = 21.

4.2.2 How Predictions Begin

The effect of updating the connection from neuron 55 to neuron 12 is first seen at the end of neuron 12's first firing burst (t = 30). Up until t = 28, neuron 12 is being fired by input 3 which stops at t = 27.

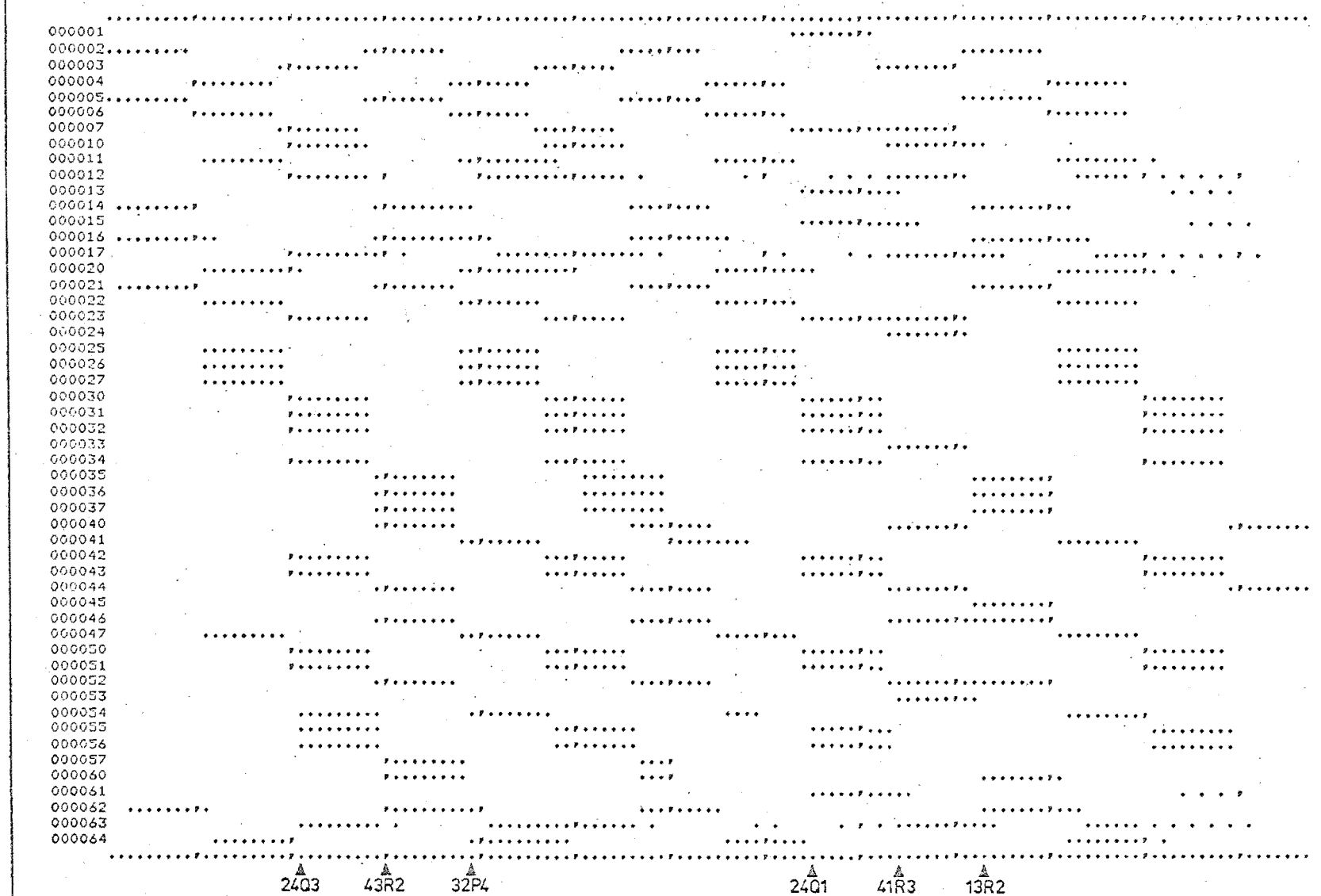


FIGURE 12. RESULTS WITH INPUT SEQUENCE A

At t = 28, context neuron 55 is still firing, but Figure 13 shows that at this time the SOB concentration of 24Q3 has died away to 73_8, so the strength of the connection to neuron 12 will be set to this value. Since this is below threshold (75_8), neuron 12 cannot fire at t = 29, but its firing potential remains at 65_8 (73_8 minus 6 die-away) (see the rules of section 1.2.1). So, when neuron 55 fires at t = 29, the firing potential of neuron 12 becomes $65_8 + 72_8 = 147_8$, which is well above threshold. Hence neuron 12 fires at t = 30 due to the context SOB code 24Q3, and its firing potential then becomes zero.

4.2.3 The Effect of a Code Match

The same integrating neurons 31, 42 and 50 are also connected to context neuron 54, and this context neuron has a potential connection to the same prediction neuron as context neuron 55, i.e. neuron 12. Hence, conditions exist for the same code as in 55, 24Q3, to be synthesised in neuron 54 at 6 = 20-21. However, because the code synthesis algorithms consider each context neuron in turn, code 24Q3 has already been completed in neuron 55 before neuron 54 is considered. Since these two neurons have potential connections to the same prediction neuron, algorithm A now checks for a code match and, finding one, rejects the code. In fact the algorithm finds a match eight times, until t = 29 when neuron 12 does not fire.

At t = 38-39, integrating neurons 25, 41 and 47 fire, followed by prediction neuron 11, context neuron 54 obtains a locally unique code 32P4.

4.2.4 Random Choice Between Connections

Context neuron 60 has two connections from the lower layer of integrating neurons: from neurons 36 and 37. If algorithm D attempts to partially synthesise a SOB code for neuron 60, it must take the input SOB stored in a lower-layer integrating neuron that has just fired as the L-part of a resulting context code. Hence a random choice must be made if there is more than one such integrating neuron, as there is here. (However, in this case there is the same SOB code 3 in both neurons 36 and 37 so the context SOB code is the same whichever of these is chosen.)

Algorithm D makes this choice several times before algorithm A can complete a code for context neuron 60. The choice is made from t = 29 - 37, t = 56 - 59, and t = 92. Algorithm A completes code 13R2 in neuron 60 at t = 93.

4.2.5 How a SOB Code Makes a Context Neuron Exclusive

Integrating neurons which have input SOBs different from the part of a context SOB code corresponding to their layer (i.e. M, L or U) have no effect on the firing potential of that context neuron if they fire into it. For example, at t = 21, neuron 54 begins a sequence of nine firing pulses. At this time it has no context SOB code and it is receiving firings from neurons 31, 42 and 50 during t = 20 - 28. However, when the same sequence of neurons 31, 42 and 50 occurs again from t = 47 to t = 55, neuron 54 does not fire. This is because it now has the context SOB code 32P4. Neuron 42 (M-layer) has input SOB 2, neuron 31 (L-layer) has SOB 4, and neuron 50 (U-layer) has Q;

none of these corresponds to its respective M, L or U part in the context SOB code.

4.2.6 The Effects of Leak-Back

Leak-back paths between context SOB codes are shown in Figure 13. The effects of leak-back can be seen in the graphs of SOB code concentration. The context SOB code 32P4, for example, was completed at t = 39 to record the fact that the context 32P was followed by the event 4. Now when the context occurs again, event 4 should be predicted. However, the strength with which it will be predicted will depend on the concentration of the context SOB code 32P4 at the time. At t = 75, the SOB code 24Q1 is completed. The event 2 has been followed by the event 4 in this context, and this could be considered as further justification for the prediction of event 4 after event 2 by code 32P4. So this 'strengthens' the prediction of 4 by 32P and the context SOB code 32P4 rises correspondingly. This increase in concentration itself causes other code concentrations to increase (it increases that of 43R2, which increases the concentration of 24Q3, and so on). Since an increase in the concentration of code 32P4 increases the strength of the connection to its prediction neuron, this causes the prediction neuron to fire more often when the context 32P occurs, and this increase in the firing rate is transmitted directly to the output. Note also that the output neurons are connected back to input neurons so that the network receives input pulses as a result of its own predictions. Thus it will tend to continue functioning for a time dependent on the threshold levels if input sequences are missed out or stop completely.

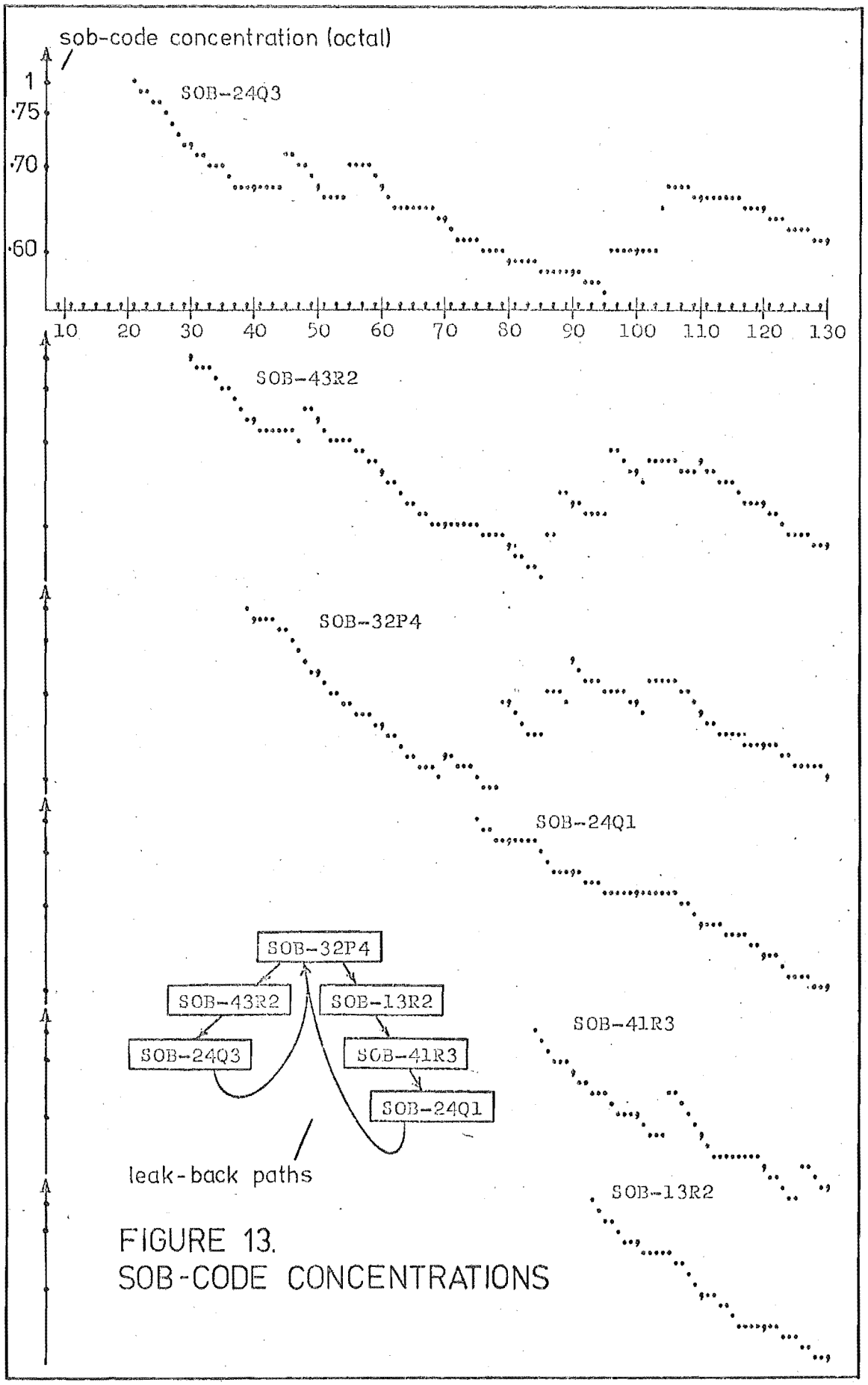


FIGURE 13.
SOB-CODE CONCENTRATIONS

Data squence A is an example where there is a wrong input pattern (t = 73 - 81). The sequence of inputs 1 - 4 is 2,4,3,2,4,3,2,4,1,3,2,4. Notice that the final prediction at the output (neurons 61 - 64) is 1 and 3 with equal strength.

The prediction of input 3 by neuron 63 between 6 = 104 and t = 111 is due to the fact that context neuron 54 has been given a potential connection to prediction neuron 12 as well as to neuron 11. In a practical network the probability of such an occurrence would be made extremely low. In this example the connection serves to illustrate the identification of a code match by the program (described in section 4.2.3).

Results appear in Figure 14 for a run using the same network structure as described already, but using input sequence B of Figure 11. Note that in this case part of the expected sequence has been omitted (lines 6 and 11). The results show a tendency for the network to 'fill in' the missing parts of the sequence and to carry on when the input sequence ends.

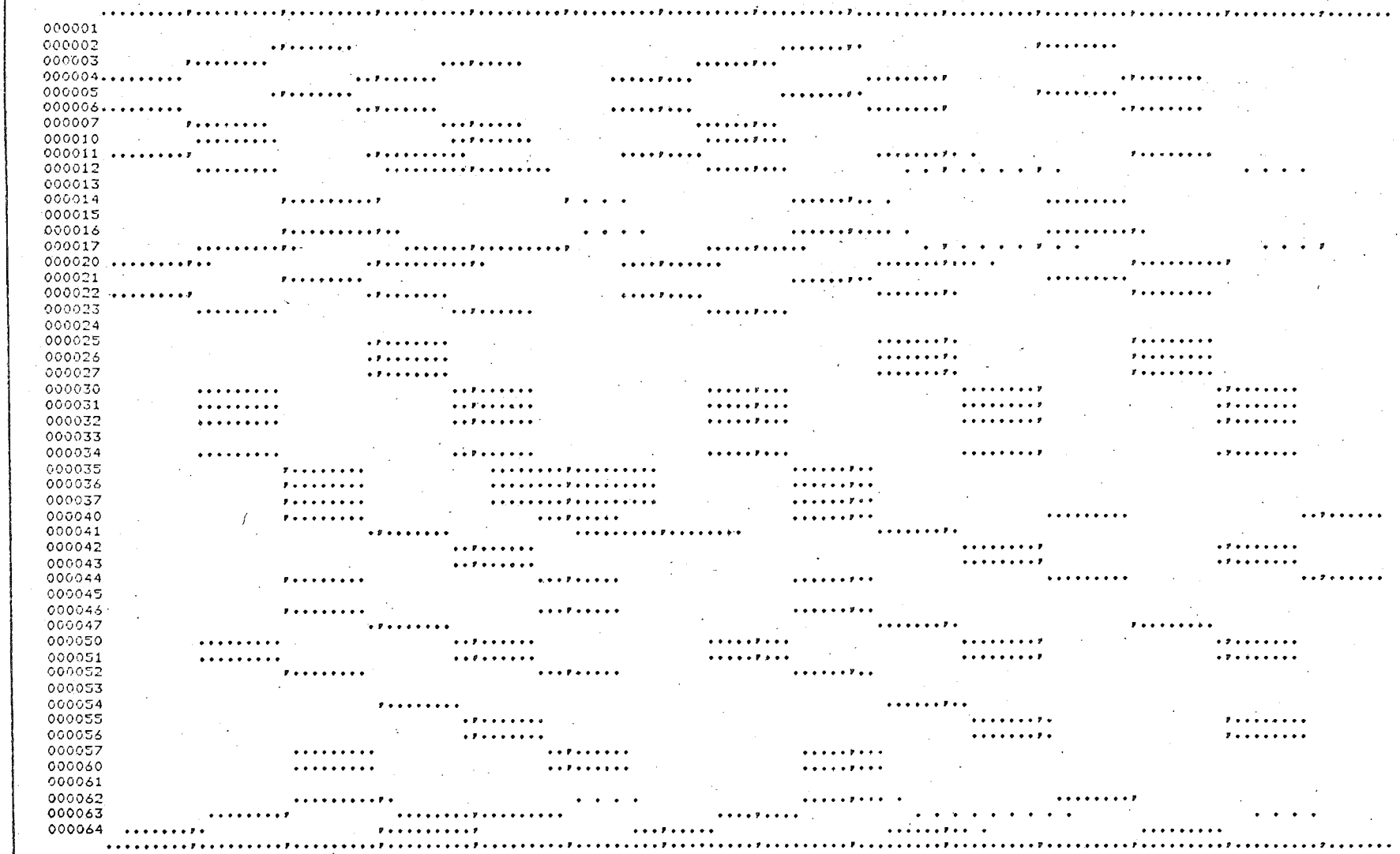


FIGURE 14. RESULTS WITH INPUT SEQUENCE B

CHAPTER 5

FUTURE DEVELOPMENTS IN MEW-BRAIN

At present, all connections between neurons in mew-Brain are specified for a particular simulation by the programmer. However, for the simulation of a network containing in the order of 1000 neurons, specifying the connections for each neuron would become extremely tedious.

A possibility in the future is the development of algorithms that would make pseudorandom connections between specified kinds of neuron in mew-Brain. Not only would this save a great deal of tedious work, but from a neuro-physiological point of view is perhaps more reasonable than having completely pre-specified connections. A particular pseudorandom function could be interrogated to specify the connections between, say, integrating neurons and context neurons, while a different function would prescribe connections between input neurons and integrating neurons. Perhaps these functions could be thought of as performing the kinds of tasks that genes do. From the point of view of computer simulation, there would be considerable savings in memory space if connections did not have to be stored.

The spatial-sequential nature of mew-Brain contexts may lead to interesting possibilities in the construction of 'contexts of contexts'. Outputs of context neurons can become inputs to another cortical-PUSS, so building up larger contexts for more sophisticated systems.

The important problem of implementing a threading PUSS using mew-Brain mechanisms is one that has not been tackled yet. In PURR-PUSS, a threading PUSS receives events only of a certain type, so that its 'window' can extend over an indefinite period of time. However, the prediction from its present context must be ready at all times. The problem in mew-Brain is that predictions begin to die away as soon as they are made. Hence the prediction from a threading 'cortical PUSS' (the name for a mew-Brain network performing the function of a single PUSS) may no longer exist when required.

Solutions to this problem have been discussed recently.* They include:

(i) storage of the prediction from a threading cortical PUSS as a chemical code;

(ii) the incorporation of reverberating circuits which would keep prediction neurons firing until new input to the cortical PUSS arrived;

(iii) the possibility that there may be no equivalent of the threading PUSS in mew-Brain.

---

*Andreae, J.H., Dowd, R.B., Webb, O.J. 'A Dual Model of the Brain' submitted to The Behavioural and Brain Sciences, March 1977.

CHAPTER 6

SUMMARY

Mew-Brain, a neurochemical version of the learning system PURR-PUSS, has been described.

An Assembler computer program has been presented which enables the simulation of a mew-Brain of approximately 1000 neurons.

The chemical coding algorithms developed enable completely general operation of mew-Brain; codes are generated and chemical leak-back occurs as a result of the structure of the network simulated and its input sequences while in operation.

Developments which are likely in the future include algorithms for the construction of pseudorandom connections, the use of higher-level contexts by feeding the outputs of one 'cortical-PUSS' into another PUSS, and an investigation of the problem of threading cortical PUSSes.

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Professor L. Kay (blind mobility project) and Dr J.H. Andreae (mew-Brain), for their encouragement and help.

I am grateful also to my parents for their continual support.

Thank you, also, A.G.K.

APPENDICES

Appendix 1.

1.1 Listing of Mew-Brain Source File

This listing has been included to facilitate the continuation of work on mew-Brain. The comments in the listing describe technical features of the program.

```
001: *
002: *
003: *
004: *
005: *
006: *
007: *
008: *
009: *
010: *
011: *
012: *
013: *        ************
014: *        *           *
015: *        * MEW-BRAIN *
016: *        *           *
017: *        ************
018: *
019: *
020: *      R.B.DOWD  MARCH '77
021: *
022: *
023: *   OBJECT FILE OF THIS PROGRAM LOADED FIRST, FOLLOWED
024: *   BY TRACO1, FOLLOWED BY OBJECT FILE OF PROGRAM 'LAST'
025: *   TO GIVE C.I. FILE MEWB.
026: *
027: *
028: *   SSW A SUPPRESSES ALL OUTPUT
029: *   SSW B PRINTS SOB CONCENTRATIONS
030: *   SSW E PRINTS MARKERS
031: *
032: *
033: *   MARKERS ARE:
034: *    EVERY FIRING CYCLE                        :
035: *    TEN FIRING CYCLES                         ;
036: *    CONTEXT SOB CODE COMPLETED                *
037: *    LOCAL CODE MATCH FOUND                    M
038: *    RANDOM CHOICE BETWEEN CONNECTIONS         0/1
039: *    SUM OF DELTA(LEAKBACK) OVERFLOWS          >
040: *    LEAKBACK INCREMENT ZERO                   <
041: *
042: *
043: *   CHANGE RATES OF DIE-AWAY AND LEAKBACK BY
044: *   CHANGING DELTA(DIE-AWAY) IN 'DELTAD' AND
045: *   DELTA(LEAKBACK) IN 'DELTAL'.
046: *
047: *
048: *   MEWB STARTS AT '1000. HAVE DATA TAPE SPECIFYING
049: *   NETWORK STRUCTURE POSITIONED IN P.T. READER
050: *   AND READER ON BEFORE STARTING. ORDER OF NEURON
051: *   DATA IS: INPUTS, TYPE 3'S, INPUT NEURONS, LOWER
052: *   LAYER TYPE 2'S,M LAYER TYPE 2'S,U LAYER TYPE2'S,
053: *   TYPE 1'S, OUTPUT NEURONS.
054: *
055: *
056:          REL       0
057:          EXTERN    BASE1,SOBASE
058: *
059: *
060: *****
061: *
062: *1. SET UP NETWORK
```

```
063: *
064: ****
065: *
066: *
067: *1.1. READS NO. OF NEURONS AND SETS SIZE OF BLOCKS
068: *  2 & 3 IN CORE. NOTE SOBASE SPECIFIED ELSEWHERE (IN
069: *  PROG. 'LAST') AND BLOCK 1 HAS SIZE 1023 WORDS. FILE
070: *  LAST ALSO SPECIFIES BASE1. LASTO (OBJECT FILE) IS
071: *  LOADED LAST SO THAT BASE1 IS POSITIONED IMMEDIATLEY
072: *  AFTER PROGRAM.
073: *
074: *
075:          L          READ1
076:          STA        NONEUS      NO. OF NEURONS
077:          ALS        1
078:          STA        TNONEU      2*NONEUS
079:          ARS        1
080:          S          TNONEU
081:          STA        MNONEU      -NONEUS
082:          A          MNONEU
083:          STA        MTWNEU      -(2*NONEUS)
084:          LA         BASE1
085:          S          MTWNEU
086:          STA        BASE2
087:          AOA
088:          STA        BAS2P1      BASE2+1
089:          S          MNONEU
090:          STA        EMPTY       FIRST EMPTY LOCATION
091:          LA         MTWNEU
092:          A          MNONEU
093:          SSP
094:          EX
095:          CLR
096: MEMCLR   STA,IX     EMPTY
097:          ICX        1
098:          J          MEMCLR
099: *1.2.FIRST EMPTY LOCATION AFTER BLOCK 3 IS 'MEMLO'
100: *  FOR SUBROUTINE MEMORY WHICH LINKS CORE INTO TWO-
101: *  WORD NODES FROM 'MEMLO' TO 'MEMHI'. THIS SPACE
102: *  (BLOCK 4) CAN THEN BE USED TO CONTAIN LISTS.
103: *  HIGHEST 'MEMHI' POSSIBLE IS 24,777 SINCE NEXT CELL
104: *  IS 'DATBAS'.
105:          LA         ME
106:          CALL       PRINT
107:          LA         ML
108:          CALL       PRINT
109:          LA         0
110:          CALL       PRINT
111:          LA         EMPTY
112:          CALL       WRITE
113: *  SUBROUTINE MEMORY ASKS FOR MEMORY AREA WITH 'MEM-'.
114: *  OPERATOR TYPES 'MEMLO,MEMHI'.
115:          CALL       MEMORY
116:          CLR
117:          EQ
118:          LA         MTWNEU
119:          SSP
120:          EX
121: *1.3.READS IN DATA FOR EACH NEURON IN TURN. SEQUENCE
122: *  IS: NEURON TYPE, LAYER IF TYPE 2, DESTINATION NEURON
123: *  +STRENGTH OF CONNECTION,...,...,...CR
124: *  FIRST READS IN NEURON TYPE:
125: NEWNEU   L          READ1
126:          LLS        14
127:          STA,IX     BAS2P1
128:          C          TYPE2
```

```
129:          SE
130:          J          CNTNU       CONTINUE
131: *  LAYER READ IF INTEGRATING NEURON.                                  56
132:          L          READ1
133:          ICX        1
134:          STA,IX     BAS2P1
135:          DCX        1
136: CNTNU    EX                      DIVIDE BY 2
137:          ARS        1
138:          OR         SETBIT
139:          EX
140:          STX        STORE
141:          LA,IX      EMPTY
142:          STA        POINTR
143: *  CONNECTION AND STRENGTH OF CONNECTION DATA IS
144: *  PUSHED ONTO A LIST IN BLOCK 3 AND THE POINTER
145: *  STORED IN BLOCK 2 FOR EACH NEURON.
146: MORCON   L          READ1
147:          CALL       PUSH,POINTR
148:          LA         POINTR
149:          LX         STORE
150:          STA,IX     EMPTY
151:          SQE                    COMMA OR CR?-STORED IN READ1
152:          J          DECREM
153:          J          MORCON      MORE CONNS.
154: DECREM   EX                     2*(X REG)
155:          ALS        1
156:          SSP
157:          EX
158:          ICX        2
159:          J          NEWNEU      NEW NEURON
160: *
161: *
162: *****
163: *
164: *2. LEAKAGE PATHS FOR INPUT SOBS
165: *
166: *****
167: *
168: *2.1.LEAKS FROM SUCCESSIVE INPUTS (SPECIFIED AS
169: *  INPUT NEURONS) TO FIRST NEURON CONNECTED SO LONG
170: *  AS IT IS NOT A CONTEXT NEURON. GOES TO 2.2. IF
171: *  OTHER THAN INPUT NEURON IS NEXT LEAKER.
172:          LX         C1
173: INPNL    LA,IX      BASE1
174:          AND        TESTYP
175:          C          TYPE0
176:          SE
177:          J          FINISH
178:          STX        KEEPX
179:          ICX        1           (X-REG+1)/2
180:          EX
181:          ARS        1
182:          EX
183:          LA,IX      BASE2
184:          STA        POINTR
185:          EX
186:          LLS        2
187:          STA        INPSOB      INPUT SOB NO
188: RECSOB   CALL       SCAN,POINTR,DONE
189:          LRS        6
190:          ALS        1
191:          EX                     (2*DEST. NEURON)
192:          DCX        1           CHECK NOT CONTEXT NEURON
193:          LA,IX      BASE1
194:          AND        TESTYP
```

```
195:          C           TYPE1
196:          SNE
197:          J           RECSOB
198:          ICX         1
199:          LA,IX       BASE1
200:          OR          INPSOB
201:          STA,IX      BASE1
202:          J           RECSOB      RECORD SOBS
203: DONE     LX          KEEPX
204:          ICX         2
205:          J           INPNL       INPUT NEURON LOOP
206: *2.2.PREDICTION NEURONS NEXT IN ORDER. DOES NOT LEAK
207: *  TO THESE AS INPUT SOBS NEVER NEED TO BE READ FROM
208: *  THEM.
209: FINISH   ICX         2           PREDICTION NEURONS
210:          LA,IX       BASE1
211:          AND         TESTYP
212:          C           TYPE3
213:          SNE
214:          J           FINISH
215: *2.3.INPUT NEURONS (TO WHICH INPUTS ARE CONNECTED)
216: *  ARE ON DATA LIST NEXT. CAN NOW LEAK FROM THEM AS INPUT
217: *  SOB WAS STORED IN THEM IN 2.1. OTHER NEURONS FOLLOW
218: *  THESE.
219:          ICX         1           OTHER INPUT NEURONS
220: TRNSOB   LA,IX       BASE1
221:          AND         HLDSOB      HOLD SOB
222:          STA         KEEPSB      KEEP SOB
223:          STX         HOLDX
224: *  CHECK INPUT SOB NOT LEAKING TO A CONTEXT NEURON.
225:          EX
226:          ARS         1
227:          EX
228:          LA,IX       BASE2
229:          STA         SCNPTR      SCAN POINTER
230: LEASOB   CALL        SCAN,SCNPTR,THRU
231:          LRS         6
232:          ALS         1           (2*DEST. NEURON)
233:          EX
234:          DCX         1           CHECK NOT CONTEXT NEURON
235:          LA,IX       BASE1
236:          AND         TESTYP
237:          C           TYPE1
238:          SNE
239:          J           LEASOB
240:          ICX         1
241:          LA,IX       BASE1
242:          OR          KEEPSB
243:          STA,IX      BASE1
244:          J           LEASOB      LEAK SOBS
245: *  TESTS TO SEE WHETHER INPUT SOBS LEAKED FAR ENOUGH.
246: *  DOESN'T NEED TO GO PAST LOWER LAYER INTEGRATING
247: *  NEURONS AS CAN GUARANTEE THAT UPPER & MIDDLE LAYER
248: *  NEURONS WILL LEAK TO CONTEXT NEURONS. THEREFOR
249: *  JUMPS OUT WHEN FIRST M LAYER NEURON ENCOUNTERED.
250: THRU     LX          HOLDX
251:          ICX         2
252:          LA,IX       BASE1
253:          AND         TESTLA      TEST LAYER
254:          C           MLAYER      MIDDLE LAYER SOBS. GONE FAR ENOUGH
255:          SE
256:          J           TRNSOB      TRANSMIT SOBS
257: *2.4.INITIALISES FIRING CYCLES.
258:          CAO
259:          STA         STEP
260:          STA         SOBNUM
```

```
261:          LA          M12
262:          STA         TENCYS      TEN CYCLE MARKER
263:          LA          M120
264:          STA         NOMARK      NO. OF MARKERS
265: *  CLEARS AREA WHERE OUTPUT DATA IS TO BE STORED.
266:          LA          NONEUS
267:          ALS         2
268:          A           NONEUS      5*NONEUS
269:          EX
270:          CLR
271: CLRCOR   STA,IX      DATBAS
272:          DCX         1
273:          J           CLRCOR      CLEAR CORE
274: *
275: *
276: *****
277: *
278: *3.  INPUT STAGE
279: *
280: *****
281: *
282: *  WHEN PROGRAM PRINTS 'INPUT-' AND PAUSES, POSITION
283: *  INPUT SEQUENCE TAPE IN P.T. READER AND R-S-R.
284: *  SEQUENCE FORMAT IS: NO. OF PULSES,NEURON NO.+STRENGTH
285: *  OF INPUT,,,,,, ....CR.
286:          LA          IN
287:          CALL        PRINT
288:          LA          PU
289:          CALL        PRINT
290:          LA          T_
291:          CALL        PRINT
292:          LA          CRLF
293:          CALL        PRINT
294:          P           '3
295: *3.1.DECIDES WHETHER NEW SEQUENCE REQUIRED.
296: INPUT    LA          PULSES
297:          S           C1
298:          STA         PULSES
299:          C           C0
300:          SG
301:          J           NEWINP      NEW INPUT
302: *3.2.SEARCHS FOR STORED INPUT SEQUENCE DATA (BITS2-7
303: *  OF WORD 1 FOR EACH NEURON IN BLOCK 2) AND UPDATES
304: *  FIRING POTENTIAL OF THOSE NEURONS WITH INPUT DATA.
305:          LA          MTWNEU
306:          SSP
307:          EX
308: SEARCH   LA,IX       BAS2P1
309:          AND         KEEPIN      KEEP INPUT DATA
310:          C           C0
311:          SG
312:          J           INCRX
313:          LRS         8
314:          STA         HOLDIN      HOLD INPUT
315:          LA,IX       BAS2P1
316:          AND         TESTYP
317:          C           TYPE1
318:          SNE
319:          J           INCRX
320:          LA,IX       BAS2P1
321:          A           HOLDIN
322:          STA,IX      BAS2P1
323: INCRX    ICX         2
324:          J           SEARCH
325:          J           INCREM
326: *3.3.READS NEW INPUT SEQUENCE FROM P.T. CLEARS OLD
```

```
327: *  SEQUENCE FIRST.
328: NEWINP  LA         MTWNEU
329:         SSP
330:         EX
331: CLEOLD  LA,IX      BAS2P1
332:         AND        TESTYP
333:         C          TYPE1
334:         SNE
335:         J          MISCON
336:         LA,IX      BAS2P1
337:         AND        ZEROIN      ZERO INPUT
338:         STA,IX     BAS2P1
339: MISCON  ICX        2
340:         J          CLEOLD      CLEAR OLD INPUT
341: *  READS NO. OF PULSES OF THE NEW SEQUENCE.
342:         L          READ1       NEW INPUT SEQUENCE
343:         STA        PULSES
344: *  READS NEURON NO. + STRENGTH OF INPUT AND STORES.
345: MORDAT  L          READ1       DEST&STR DATA
346:         EQ
347:         STA        CTEST       COMMA TEST
348:         CLR
349:         LLD        10
350:         C          C0
351:         SNE
352:         J          INCREM
353:         ALS        1
354:         S          C1          (2*NO. OF NEURON)-1
355:         EX
356:         CLR
357:         LLD        6
358:         STA        HOLD
359:         LLS        8
360:         A          HOLD
361:         STA        HOLD
362:         LA,IX      BASE1
363:         A          HOLD
364:         STA,IX     BASE1
365:         LA         CTEST
366:         SAE
367:         SKU                    INCREMENT STAGE
368:         J          MORDAT      MORE DATA
369: *
370: *
371: *****
372: *
373: *4. INCREMENT STAGE
374: *
375: *****
376: *
377: *4.1.CHECKS WHETHER TIME TO PRINT COMPUTED RESULTS
378: *  (AFTER 80 CYCLES). IF NOT TIME, USES STEP NO.
379: *  TO DECIDE WHICH BIT WILL BE SET OUT OF THE 80
380: *  IN THE 5 WORDS ALLOCATED FOR EACH NEURON, IF THAT
381: *  NEURON FIRES THIS CYCLE.
382: INCREM  CLR
383:         EQ
384:         LA         STEP
385:         S          C1
386:         LRD        4
387:         C          C5
388:         SL
389:         J          OUTPUT      PRINT OUTPUT
390:         STA        WRDNUM      WORD NUMBER
391: *  BIT SET WILL BE IN THIS ONE OF THE 5 WORDS.
392:         CLR
```

```
393:          LLD       4
394:          AOA
395:          STA       SHIFT1
396: *   THE REQUIRED BIT WILL BE POSITIONED IN BIT 15
397: *   OF THE ACCUM. IF THE WORD CHOSEN ABOVE IS SHIFTED
398: *   'SHIFT1' PLACES FROM THE Q-REGISTER.
399:          LA        C20
400:          S         SHIFT1
401:          STA       SHIFT2
402: *   THE WORD WILL THEN NEED TO BE SHIFTED 'SHIFT2'
403: *   PLACES TO RETURN IT ENTIRELY TO THE ACCUM.
404:          AOM       STEP
405:          LX        NONEUS
406: *4.2.INCREMENTS FIRING POTENTIALS OF DESTINATION
407: *   NEURONS IF SOURCE NEURON FIRED.(BIT 0 IN CELLS
408: *   OF BLOCK 3 SET IF NEURON FIRED IN LAST CYCLE).
409: *   FIRST CHECKS IF NEURON FIRED AND RESETS BIT IF SO.
410: INLOOP   LA,IX      BASE2
411:          SKN
412:          J         DECRX
413:          SSP
414:          STA,IX    BASE2
415:          STA       POINTR          STORE SCAN POINTER
416:          STX       KEEPX
417: *   FINDS NEURONS TO WHICH IT IS CONNECTED AND
418: *   UPDATES THEIR FIRING POTENTIALS.
419: SCLOOP   CALL      SCAN,POINTR,NOCONS
420:          EQ
421:          CLR
422:          LLD       10
423:          ALS       1
424:          EX
425:          CLR
426:          LLD       6
427:          STA       STOREA
428: *4.2.1.DOES NEURON CONNECTED HAVE A COMPLETE
429: *   CONTEXT SOB CODE?
430:          LA,IX     BASE1
431:          DCX       1
432:          SKP
433:          J         CONTST      CONTEXT TEST
434: *   NO. CONTINUE AS NORMAL.
435: FALSE    LA,IX     BASE1
436:          A         STOREA
437:          STA,IX    BASE1
438:          J         SCLOOP      SCAN LOOP
439: *   THIS PART OF THE ROUTINE STOPS UPDATE OF
440: *   FIRING POTENTIAL FROM INTEGRATING NEURON
441: *   TO CONTEXT NEURON WITH COMPLETE SOB CODE
442: *   IF INPUT SOB IN INTEGRATING NEURON IS NOT
443: *   THE SAME AS PART OF SOB CODE IN CONTEXT
444: *   NEURON CORRESPONDING TO INTEGRATING
445: *   NEURON'S LAYER.
446: *   FIRST CHECKS IT IS CONTEXT NEURON.
447: CONTST   LA,IX     BASE1
448:          AND       TESTYP
449:          C         TYPE1
450:          SE
451:          J         FALSE
452: *   STORES SOB ADDRESS FOR THIS CONTEXT NEURON.
453:          ICX       1
454:          LA,IX     BASE1
455:          AND       KPADRS
456:          STA       CONSAD      CONSTRUCT ADDRESS
457:          DCX       1
458:          STX       HOLDX       NEURON CONNECTED TO
```

```
459:          LA          KEEPX       CONNECTER
460:          ALS         1
461:          S           C1
462:          EX
463: *   CHECKS CONNECTER IS INTEGRATING.
464:          LA,IX       BASE1
465:          AND         TESTYP
466:          C           TYPE2
467:          SE
468:          J           INCONT      INCREMENT CONTINUE
469:          ICX         1
470: *   STORES INPUT SOB.
471:          LA,IX       BASE1
472:          EQ
473:          LLD         6
474:          CLR
475:          LLD         8
476:          STA         INPSOB      INPUT SOB
477: *   CONSTRUCTS ADDRESS.
478:          CLR
479:          LLD         2
480: *   CHANGES 1-0,2-1,3-2 AND POSITIONS RESULT ABOVE
481: *   ADDRESS BITS.
482:          C           C3
483:          SNE
484:          A           C1
485:          LRS         1
486:          LLS         8
487:          A           CONSAD
488:          EX
489:          LA,IX       SOBASE
490:          C           INPSOB
491:          SE
492:          J           SCLOOP
493: INCONT   LX          HOLDX
494:          J           FALSE
495: NOCONS   LX          KEEPX
496: DECRX    DCX         1
497:          J           INLOOP      INCREMENT LOOP
498:          AOM         TENCYS
499: *   PRINTS MARKER EVERY CYCLE.
500:          J           PCOLON      PRINT COLON
501: *   PRINTS MARKER FOR EVERY 10 CYCLES.
502:          LA          M12
503:          STA         TENCYS
504:          LA          SMICLN      TEN CYCLES
505:          L           MARKER
506:          J           FIRING
507: PCOLON   LA          COLON       CYCLE MARKER
508:          L           MARKER
509:          J           FIRING
510: *
511: *
512: *4.3.PRINTS COMPUTED DATA IN GRAPHIC DISPLAY.
513: *   PAUSES BEFORE OUTPUT FOR POSITIONING OF DECWRITER
514: *   PAPER AND/OR CHANGING SSW A STATUS.
515: OUTPUT   F           '10
516:          SSW         A
517:          SKU
518:          J           NONOUT      NON OUTPUT
519:          LA          CRLF
520:          CALL        PRINT
521: *   MARKS OFF HORIZONTAL AXIS.
522:          L           DOTTED
523:          LA          C11
524:          STA         MARKS
```

```
525: *  'DATBAS' IS BEGINNING OF OUTPUT DATA BLOCK
526: *  IN CORE.
527:          LA         DATBAS
528:          S          C1
529:          STA        START
530:          CAO
531:          STA        NEURNO
532: *  NEW LINE AND PRINTS NEURON NO. ON VERTICAL AXIS.
533: NXTNEU   LA         CRLF
534:          CALL       PRINT
535:          LA         NEURNO
536:          CALL       WRITE
537:          LA         N5
538:          STA        WORDS
539: *  EXAMINES EACH WORD AFTER 'DATBAS' IN TURN,PRINTING
540: *  A DOT FOR EVERY POSITION FOR WHICH THERE IS A
541: *  LOGICAL 1.(PRINTS COMMA FOR EVERY 10 DOTS OR BLANKS).
542: NEWWRD   AOM        START
543:          LA,I       START
544:          STA        KEPT
545:          LX         C17
546: DLOOP    LA         KEPT
547:          EQ
548:          CLR
549:          LLD        1
550:          EQ
551:          STA        KEPT
552:          EQ
553:          C          C1
554:          SE
555:          J          BLANK
556:          LA         MARKS
557:          C          C0
558:          SNE
559:          J          MARK
560:          LA         DOT
561:          CALL       PRINT
562:          J          BACK
563: MARK     LA         C254
564:          CALL       PRINT
565: BACK     LA         MARKS
566:          C          C0
567:          SG
568:          LA         C12
569:          S          C1
570:          STA        MARKS
571:          DCX        1
572:          J          DLOOP
573:          CLR
574:          STA,I      START
575:          AOM        WORDS
576:          J          NEWWRD
577: *  CHECKS THAT HIGHEST NUMBERED NEURON NOT EXCEEDED.
578:          LA         NEURNO
579:          C          NONEUS
580:          SL
581:          J          CMPUT
582:          AOM        NEURNO
583:          J          NXTNEU
584: BLANK    LA         SPACE
585:          CALL       PRINT
586:          J          BACK
587: *  HORIZONTAL AXIS AGAIN. INITIALISES STEP AND
588: *  RETURNS TO COMPUTATION AFTER PAUSE.
589: CMPUT    LA         CRLF
590:          CALL       PRINT
```

```
591:          L           DOTTED
592:          LA          CRLF
593:          CALL        PRINT
594:          P           '7
595: NONOUT   CAO
596:          STA         STEP
597:          LA          M120
598:          STA         NOMARK
599:          J           INCREM
600:
600: *
002: *
003: *****
004: *
005: *5. FIRING STAGE
006: *
007: *****
008: *
009: FIRING   LA          NTWNEU
010:          SSP
011:          EX
012: FRLOOP   CLR
013:          EQ
014: *   JUMPS TO 'ZEROFC' FOR NON-INTEGRATING NEURONS.
015:          LA,IX       BAS2P1
016:          SKN
017:          J           ZEROFC      ZERO F.C. NEURON
018:          ICX         1
019: *   THESE ARE INTEGRATING NEURONS. CHECKS IF THEY HAVE
020: *   A FIRING COUNT AND GOES TO 'ZEROFC' IF NOT. 'NZERFC'
021: *   DECREASES THEIR FIRING COUNT BY 1 IF NOT ZERO, THEN
022: *   THEY WILL FIRE.
023:          LA,IX       BAS2P1
024:          EQ
025:          LLD         6
026:          C           C0
027:          SE
028:          J           NZERFC      NONZERO FIRING COUNT
029:          DCX         1
030:          CLR
031:          EQ
032:          LA,IX       BAS2P1
033:          J           ZEROFC
034: NZERFC   S           C1
035:          LLD         10
036:          STA,IX      BAS2P1
037:          DCX         1
038: *   NEURON WILL DEFINATELY FIRE.
039:          J           FIRE
040: ZEROFC   EQ
041:          LLD         8
042:          EQ
043:          LRS         8
044: *   ANY NEURON WITH ZERO FIRING COUNT FIRES IF ITS
045: *   FIRING POTENTIAL IS ABOVE THRESHOLD.
046:          C           THRESH
047:          SG
048:          J           TFPRED      TEST FOR F.P. REDUCE NON FIRE
049: *   NEURON WILL DEFINATELY FIRE. ALSO TESTS FOR
050: *   INTEGRATING NEURON AND SETS INITIAL COUNT IF ONE
051: *   FOUND.
052: *   IN BOTH CASES FIRING POTENTIAL SET TO ZERO.
053:          EQ
054:          LRS         6
055:          C           C2
056:          SE
```

```
057:          J          FPOFIR      F.P.=0 & FIRE
058:          ICX        1           INTEGRATION FIRING COUNT
059:          LA,IX      BAS2P1                                  64
060:          A          INTIC       INT. INITIAL COUNT
061:          STA,IX     BAS2P1
062:          DCX        1
063: FPOFIR   LA,IX      BAS2P1
064:          AND        FPZERO      ZERO F.P.
065:          STA,IX     BAS2P1
066: FIRE     STX        STOREX
067: *   CHANGES NO. IN X-REGISTER INTO (NEURON FIRED-1).
068:          EX
069:          SSN
070:          ARS        1
071:          A          NONEUS
072:          STA        NFIRED      NEURON FIRED - 1
073: *   'DATBAS' IS THE DATA BASE- LOWEST CELL OF BLOCK
074: *   IN CORE IN WHICH OUTPUT DATA STORED. WHEN A
075: *   NEURON FIRES,APPROPRIATE BIT OF APPROPRIATE WORD
076: *   IS SET. 'DATBAS'+[5*(NEURON FIRED-1)] GIVES
077: *   ADDRESS OF BEGINNING OF 5-WORD BLOCK FOR THIS
078: *   NEURON. ADDS TO THIS 'WRDNUM'(EITHER 1,2,3,OR 4),
079: *   FOUND IN 4.1 OF THIS CYCLE, TO GIVE ADDRESS OF
080: *   WORD IN WHICH BIT WILL BE SET. THEN CHANGES
081: *   SHIFT INSTRUCTION LABELLED 'POSITN',USING 'SHIFT1'
082: *   FOUND IN 4.1 OF THIS CYCLE SO THAT RIGHT BIT OF
083: *   THIS WORD IS POSITIONED IN BIT 15 OF ACCUM. AND
084: *   CAN BE SET TO 1.'SHIFT2' DOES SIMILAR, TO MOVE
085: *   THE WORD COMPLETELY FROM Q-REGISTER TO ACCUM. SO
086: *   IT CAN BE STORED OTHERWISE UNCHANGED IN ADDRESS
087: *   IT CAME FROM.
088:          ALS        2           *5
089:          A          NFIRED
090:          A          DATBAS
091:          A          WRDNUM
092: *   THIS IS CORRECT WORD.
093:          STA        ADDRES
094: *   CHANGES LLD INSTRUCTION.
095:          LA         POSITN      POSITION
096:          AND        SHFTO
097:          A          SHIFT1
098:          STA        POSITN
099:          LA         MOVOUT      MOVE OUT
100:          AND        SHFTO
101:          A          SHIFT2
102:          STA        MOVOUT
103: *   POSITIONS WORD IN WHICH BIT TO BE SET.
104:          LA,I       ADDRES
105:          EQ
106:          CLR
107: POSITN   LLD        0
108:          AOA
109: MOVOUT   LLD        0
110: *   STORES IT WITH RIGHT BIT CHANGED.
111:          STA,I      ADDRES
112: *   SETS TAG IN BLOCK 3 TO SHOW NEURON FIRED THIS
113: *   CYCLE.(TAG HELD FOR ONE CYCLE ONLY).
114:          LX         NFIRED
115:          LA,IX      BAS2P1
116:          SSN
117:          STA,IX     BAS2P1
118:          LX         STOREX
119: FPRED    CLR
120:          EQ
121: *   REDUCES FIRING POTENTIAL.
122:          LA,IX      BAS2P1
```

```
123:          EQ
124:          LLD       8
125:          STA       SAVEIN      SAVE BITS
126:          EQ
127:          LRD       8
128:          C         C0
129:          SNE
130:          J         OUT
131: *  SUBTRACTS DIE-AWAY IF GREATER THAN LOWER LIMIT.
132:          S         DECAY
133:          C         LIMIT
134:          SGE
135:          CLR
136:          EQ
137:          LLD       8
138:          CLR
139:          LA        SAVEIN
140:          LLD       8
141:          STA,IX    BAS2P1
142:          J         OUT
143: TFPRED   EQ
144: *  ONLY NEURONS WITH ZERO FIRING COUNT AND FIRING
145: *  POTENTIAL LESS THAN THRESHOLD GET HERE.
146:          LRD       6
147:          C         C1
148: *  FIRING POTENTIAL REDUCED ONLY IF INTEGRATING
149: *  OR PREDICTION NEURON, OTHERWISE FIRING POTENTIAL
150: *  SET TO ZERO.
151:          SLE
152:          J         FPRED       F.P. REDUCE, NON FIRE
153:          LLD       6           F.P.=0
154:          LLS       8
155:          STA,IX    BAS2P1
156: OUT      ICX       2
157:          J         FRLOOP      FIRING LOOP
158: *
159: *
160: *******
161: **  FOR FOLLOWING FOUR SECTIONS ALL COMMENT NUMBERING
162: **  FOLLOWS THAT OF ALGORITHMS IN TEXT.
163: *******
164: *
165: *
166: *
167: *****
168: *
169: *6. ALGORITHM A: CONTEXT SOB CODE COMPLETION.
170: *
171: *****
172: *
173: *A.THIS ALGORITHM ATTEMPTS TO FIND THE 'P' PART OF
174: *  A CHEMICAL CODE AND HENCE COMPLETE IT. CONTEXT
175: *  NEURONS FOR WHICH A CONTEXT FIRED THIS CYCLE ARE
176: *  EXAMINED TO SEE IF M,L,AND U PARTS OF A POSSIBLE
177: *  CODE WERE FOUND FOR THEM IN ALGORITHM D OF THE
178: *  LAST CYCLE. IF SO,THE CODE FOR THIS CONTEXT
179: *  NEURON MAY BE COMPLETED HERE. THE CODE IS COMPLETED
180: *  IF A PREDICTION NEURON TO WHICH THE CONTEXT NEURON
181: *  HAS A POTENTIAL CONNECTION ALSO FIRES THIS CYCLE,
182: *  AND IF THE COMPLETE CODE OBTAINED IS LOCALLY UNIQUE.
183:          LX        NONEUS
184: *A.1.CHECKS NEURON N JUST FIRED.
185: CODCOM   LA,IX     BASE2
186:          SKN                   FIND NEURON JUST FIRED
187:          J         DECRX1
188:          SSP
```

```
189:          STA         STPNTR
190:          STX         HOLDX
191: *A.2.CHECKS N IS CONTEXT NEURON.                                    66
192:          EX
193:          ALS         1
194:          S           C1
195:          EX
196:          LA,IX       BASE1       IS IT CONTEXT?
197:          AND         TESTYP
198:          C           TYPE1
199:          SE
200:          J           RETRY
201: *A.3.CHECKS N HAS M,L,AND U TAGS.
202:          ICX         1
203:          LA,IX       BASE1
204:          SKP                     NO P-TAG?
205:          J           RETRY
206:          AND         KPTAGS      KEEP TAGS
207:          C           MLUTAG
208:          SE
209:          J           RETRY
210:          STX         CONTX       FOR CONTEXT NEURON
211: *A.4.SCANS CONNECTIONS OF N.
212: SCAND    CALL        SCAN,STPNTR,RETRY
213: *  RETURNS WITH C(E).
214: *A.4.1.N CONNECTED TO PREDICTION NEURON P?
215:          LRS         6
216:          STA         CNECTD
217:          ALS         1
218:          S           C1
219:          EX
220:          LA,IX       BASE1       IS IT PREDICTION?
221:          AND         TESTYP
222:          C           TYPE3
223:          SE
224:          J           SCAND
225:          ICX         1
226:          LA,IX       BASE1
227:          AND         HLDSOB
228:          LRS         2
229:          STA         KEEPSB      KEEP INPUT SOB
230: *A.4.2.HAS P JUST FIRED?
231:          LX          CNECTD
232:          LA,IX       BASE2
233:          SKN
234:          J           SCAND
235: *A.4.3.STORES P-PART OF CONTEXT SOB CODE.
236:          LX          CONTX
237:          LA,IX       BASE1
238:          AND         KPADRS
239:          STA         HLDADR      HOLD PRESENT ADDRESS
240:          OR          PBLOCK
241:          EX
242:          LA          KEEPSB
243:          STA,IX      SOBASE
244: *A.4.4.CHECKS LOCAL UNIQUENESS. THIS MEANS CONSIDERS
245: *  WHETHER SAME CODE AS FOUND HERE IS STORED IN
246: *  ANOTHER CONTEXT NEURON M WITH POTENTIAL CONNECTION
247: *  TO SAME PREDICTION NEURON P CONSIDERED HERE.
248:          LX          TNONEU
249: UNIQUE   EX
250:          C           CONTX
251:          SNE
252: *  MISSES NEURON N.
253:          S           C2
254:          EX
```

```
255: *A.4.4.2.CHECKS NEURON M HAS P-TAG.
256:          LA,IX     BASE1       P-TAG SET?
257:          SKN
258:          J         DECRX3
259:          DCX       1
260: *A.4.4.1.CHECKS M IS CONTEXT NEURON.
261:          LA,IX     BASE1
262:          AND       TESTYP      CHECK CONTEXT
263:          ICX       1
264:          C         TYPE1
265:          SE
266:          J         DECRX3
267:          STX       STOREX
268: *A.4.4.3.SEARCHS FOR CONNECTIONS TO NEURON P.
269:          EX
270:          ARS       1
271:          EX
272:          LA,IX     BASE2
273:          SSP
274:          STA       STPNTR
275: FNDPRD   CALL      SCAN,STPNTR,RETURN
276: *  RETURNS WITH C(F).
277: *  CONTINUES ONLY IF A NEURON CONNECTED TO M IS P.
278:          LRS       6
279:          C         CNECTD
280:          SE
281:          J         FNDPRD      FIND PREDICTION NEURON
282: *A.4.4.4.COMPARES CODES OF N AND M.
283:          LX        STOREX
284:          LA,IX     BASE1
285:          AND       KPADRS
286:          STA       CMPADR      COMPARED ADDRESS
287:          EX
288:          LA,IX     SOBASE
289:          STA       COMPRD
290:          LA        HLDADR
291:          EX
292:          LA,IX     SOBASE
293:          STA       HELD
294:          C         COMPRD
295:          SE
296:          J         RETURN      UNIQUE SO FAR
297: *  M-PARTS OF CODES MATCH.
298: UNLOOP   LA        CMPADR
299:          A         C400
300:          STA       CMPADR
301:          EX
302:          LA,IX     SOBASE
303:          STA       COMPRD
304:          LA        HLDADR
305:          A         C400
306:          STA       HLDADR
307:          EX
308:          LA,IX     SOBASE
309:          STA       HELD
310:          C         COMPRD
311:          SE
312:          J         RETURN      UNIQUE SO FAR
313:          LA        CMPADR
314:          C         C1400        TESTED FAR ENOUGH?
315:          SGE
316:          J         UNLOOP      UNIQUE TEST LOOP
317: *  HAS FOUND MATCH BETWEEN M,L,U AND P PARTS.
318:          LA        M           CODE MATCH
319:          L         MARKER
320:          J         SCAND
```

```
321: RETURN  LX        STOREX
322: DECRX3  DCX       2
323:         J         UNIQUE
324: *A.4.5.SETS P-TAG AND INITIAL SOB CODE CONCENTRATION.
325:         LX        CONTX
326:         LA,IX     BASE1
327:         AND       KEEPU       SET INITIAL SOB CONC
328:         SSN                   SET P-TAG.
329:         STA,IX    BASE1
330:         LA        STAR
331:         L         MARKER
332: RETRY   LX        HOLDX
333: DECRX1  DCX       1
334:         J         CODCOM      CODE COMPLETION
335: *
336: *
337: *****
338: *
339: *7. ALGORITHM B: WIPE TAGS FOR CODES NOT COMPLETED
340: *
341: *****
342: *
343: *B.IF CODE FOR A NEURON N FAILED TO BE COMPLETED IN
344: *  ALGORITHM A, THEN ANY OF M,L,AND U TAGS SET IN
345: *  LAST FIRING CYCLE MUST BE WIPED TO ALLOW NEW
346: *  ATTEMPT AT SYNTHESIS IN ALGORITHM D THIS CYCLE.
347:         LX        TNONEU
348: *B.1.CHECKS N IS CONTEXT NEURON.
349: WIPTAG  DCX       1
350:         LA,IX     BASE1
351:         ICX       1
352:         AND       TESTYP
353:         C         TYPE1
354:         SE
355:         J         DECRX4
356: *B.2.CHECKS NO P-TAG.
357:         LA,IX     BASE1
358:         SKP
359:         J         DECRX4
360: *B.3.WIPES TAGS.
361:         AND       KPREST      KEEP REST
362:         STA,IX    BASE1
363: DECRX4  DCX       2
364:         J         WIPTAG      WIPE TAGS
365:
001: *
002: *
003: *****
004: *
005: *8. ALGORITHM C: LEAK-BACK OF CODES
006: *
007: *****
008: *
009: *C.1.THIS PART OF THE ALGORITHM FINDS LEAK-BACK
010: *  PATHS FOR EACH NEURON N AND THE AMOUNT OF THE
011: *  LEAK-BACK FOR EACH PATH FOUND.
012:         LX        TNONEU
013: *C.1.2.CHECKS P-TAG SET.
014: LEAKER  LA,IX     BASE1
015:         SKN
016:         J         DECRX7
017:         STX       HOLDX       LEAKER
018:         DCX       1
019: *C.1.1.CHECKS N IS CONTEXT NEURON.
020:         LA,IX     BASE1
021:         ICX       1
```

```
022:          AND         TESTYP
023:          C           TYPE1
024:          SE
025:          J           DECRX7
026: *C.1.3.FINDS NEURON M TO WHICH LEAKBACK PATH EXISTS.
027:          LX          JNONEU
028: *C.1.3.2.CHECKS M HAS COMPLETED CODE.
029: LEAKEE   LA,IX       BASE1
030:          SKN
031:          J           DECRX6
032:          STX         KEEPX
033: *C.1.3.1.CHECKS M IS CONTEXT NEURON.
034:          DCX         1
035:          LA,IX       BASE1
036:          ICX         1
037:          AND         TESTYP
038:          C           TYPE1
039:          SE
040:          J           DECRX6
041: *   CHECKS M NOT EQUAL TO N.
042:          EX
043:          C           HOLDX
044:          SNE
045:          J           NOMTCH
046:          EX
047: *C.1.3.3.IS THERE AN L-P MATCH?
048:          LA,IX       BASE1
049:          AND         KPADRS
050:          STA         HELD        LEAKEE BASIC ADR.
051: *   FINDS P-PART OF CODE FOR M.
052:          OR          PBLOCK
053:          EX
054:          LA,IX       SOBASE
055:          STA         COMPRD      LEAKEE P-PART
056:          LX          HOLDX
057:          LA,IX       BASE1
058:          AND         KPADRS
059:          STA         KEPT        LEAKER BASIC ADR.
060: *   FINDS L-PART OF CODE FOR N.
061:          OR          LBLOCK
062:          EX
063:          LA,IX       SOBASE
064:          C           COMPRD
065:          SE
066:          J           NOMTCH      NO L-P MATCH
067: *C.1.3.4.IS THERE ALSO AN M-L MATCH?
068:          LA          HELD
069: *   FINDS L-PART OF CODE FOR N.
070:          OR          LBLOCK
071:          EX
072:          LA,IX       SOBASE
073:          STA         COMPRD      LEAKEE L PART
074:          LX          KEPT
075:          LA,IX       SOBASE
076:          C           COMPRD
077:          SE
078:          J           NOMTCH      NO M-L MATCH
079: *   POSSIBLE PATH FOUND.
080: *C.1.3.5.CHECKS DELTA(CONC)>0.
081:          LX          KEEPX
082:          LA,IX       BASE1
083:          AND         KEEPCN
084:          STA         HOLD
085:          LX          HOLDX
086:          LA,IX       BASE1
087:          AND         KEEPCN
```

```
088:          S           HOLD
089:          C           CO
090:          SG
091:          J           NOMTCH
092:          LRS         8
093:          STA         HOLD        [CONC(LEAKER)-CONC(LEAKEE)]
094: *   'PROBABILISTIC LEAKBACK',ADDS DELTA(LEAKBACK)
095: *   IF SAMPLE OF A RANDOM NO.<'HOLD',
096: *C.1.3.6.ADD DELTA(LEAKBACK) TO PRESENT SUM OF
097: *   DELTA(LEAKBACK)?
098:          CLR
099:          STA         STORE
100: *   ATTEMPTS FIVE SMALL INCREMENTS.
101:          LA          N5
102:          STA         HOLDIN
103: DELOOP   L           RANDOM
104:          AND         RANGE
105:          C           HOLD
106:          SL
107:          J           UPWDS       NO UPDATE
108: *   DECREASES SIZE OF 'HOLD'.
109:          LA          HOLD
110:          C           DELTAL
111:          SG
112:          J           CNTNU1
113:          S           DELTAL      UPDATE BY DELTAL
114:          STA         HOLD
115: *   'STORE' INCREASES BY DELTAL.
116:          LA          STORE
117:          A           DELTAL
118:          STA         STORE
119: UPWDS    AOM         HOLDIN
120:          J           DELOOP      DELTA LOOP
121:          J           CNTNU3
122: *   JUMPS OUT AS 'HOLD' WOULD BE ZERO NEXT TIME.
123: CNTNU1   A           STORE       UPDATE BY HOLD (<DELTAL)
124:          STA         STORE
125: *C.1.3.7.ADDS TO PRESENT SUM OF DELTA(LEAKBACK).
126: CNTNU3   LX          KEEPX
127:          DCX         1
128:          LA,IX       BASE1
129:          EQ
130:          LLD         2
131:          CLR
132:          LLD         6
133:          A           STORE       LEAKBACK
134: *   PRINTS MARKER IF SUM OF DELTA(LEAKBACK)
135: *   OVERFLOWS.
136:          C           C77
137:          SG
138:          J           CNTNU2
139:          LA          GTRTHN      >
140:          L           MARKER
141:          LA          C77
142: CNTNU2   LLD         8
143:          OR          TYPE1
144:          STA,IX      BASE1
145: NOMTCH   LX          KEEPX
146: DECRX6   DCX         2
147:          J           LEAKEE
148:          LX          HOLDX
149: DECRX7   DCX         2
150:          J           LEAKER
151: *C.2.THIS PART OF THE ALGORITHM UPDATES THE SOB-CODE
152: *   CONCENTRATION OF N BY THE AMOUNT IN SUM OF DELTA
153: *   (LEAKBACK) AND UPDATES CONTEXT NEURON - PREDICTION
```

```
154:  *  NEURON STRENGTH OF CONNECTION.
155:           LX          TNONEU
156:  *C.2.2.CHECKS N HAS COMPLETED CODE.                                    71
157:  CONCUP   LA,IX       BASE1       CONCENTRATION UPDATE
158:           SKN
159:           J           DECRX8
160:           DCX         1
161:  *C.2.1.CHECKS N IS CONTEXT NEURON.
162:           LA,IX       BASE1
163:           AND         TESTYP
164:           C           TYPE1
165:           SE
166:           J           DECRX8
167:           LA,IX       BASE1
168:           EQ
169:           LLD         2
170:           CLR
171:           LLD         6
172:  *  PRINTS MARKER IF SUM OF DELTA(LEAKBACK) FOR THIS
173:  *  CYCLE IS ZERO.
174:           C           C0
175:           SE
176:           J           INRNGE
177:           LA          LESTHN      <
178:           L           MARKER
179:           CLR
180:  *  STORES SUM OF DELTA(LEAKBACK).
181:  INRNGE   STA         STOREA
182:           CLR
183:           LLD         8
184:           OR          TYPE1
185:           STA,IX      BASE1
186:           ICX         1
187:  *C.2.3.UPDATES SOB CODE CONCENTRATION.
188:           LA,IX       BASE1
189:           SSP
190:           EQ
191:           CLR
192:           LLD         8
193:           A           STOREA
194:           STA         SAVEIN      UPDATED CONC.
195:  *  'PROBABILISTIC DIE-AWAY' SUBTRACT DELTA(DIE-AWAY)
196:  *  IF SAMPLE OF RANDOM NO.< CONCENTRATION.
197:  *  SUBTRACT DELTA(DIE-AWAY) FROM UPDATED CONCENTRATION?
198:           L           RANDOM
199:           AND         RANGE
200:           C           SAVEIN
201:           SL
202:           J           CNTNU4
203:           LA          SAVEIN
204:  *  SUBTRACTS DELTA(DIE-AWAY).
205:           S           DELTAD      DIE-AWAY
206:           STA         SAVEIN
207:  CNTNU4   LA          SAVEIN
208:           LLD         8
209:           SSN                     P-TAG REPLACED
210:           STA,IX      BASE1
211:  *C.2.4.UPDATES STRENGTH OF CONNECTION TO PREDICTION
212:  *  NEURON.
213:           LA          SAVEIN
214:  *C.2.4.2.CHECKS MAGNITUDE OF SOB-CODE CONCENTRATION.
215:           C           C100
216:           SL
217:           LA          C77
218:           STA         SAVEIN
219:           STX         CONTX
```

```
220:          EX
221:          ARS         1
222:          EX
223: *  FINDS POINTER AND SCANS CONNECTIONS OF N.
224:          LA,IX       BASE2
225:          SSP
226:          STA         STPNTR
227: FINDPR   LA          STPNTR      CHANGE STRS OF CONN.
228:          AND         ZERBT1      ZERO BIT 1
229:          STA         HOLDIN      HOLD OLD POINTER
230:          CALL        SCAN,STPNTR,UPDATD
231: *  RETURNS WITH C(E).
232: *C.2.4.1.CHECKS N CONNECTED TO PREDICTION NEURON.
233:          LRS         6
234:          ALS         1
235:          S           C1
236:          EX
237:          LA,IX       BASE1
238:          AND         TESTYP
239:          C           TYPE3
240:          SE
241:          J           FINDPR      FIND PRED NEURON
242: *C.2.4.3.COPIES SOB CONCENTRATION INTO STRENGTH OF
243: *  CONNECTION.
244:          LA          HOLDIN
245:          A           C1
246:          STA         HOLDIN
247: *  'HOLDIN' NOW HAS ADDRESS OF WORD NEXT TO OLD POINTER
248: *  I.E.INFORMATION WORD REQUIRED.
249: *  NOW CHANGES CONTENTS OF HOLDIN.
250:          LA,I        HOLDIN
251:          AND         ZERSTR      ZERO STR. OF CONN.
252:          A           SAVEIN
253:          STA,I       HOLDIN
254:          J           FINDPR
255: UPDATD   LX          CONTX
256: DECRX8   DCX         2
257:          J           CONCUP      CONCENTRATION UPDATE
258:
259: *
260: *
261: ****
262: *
263: *9. ALGORITHM D:SYNTHESIS OF NEW CODES
264: *
265: ****
266: *
267: *D.IN THIS ALGORITHM,M,L,AND U PARTS OF A POSSIBLE
268: *  NEW CONTEXT SOB CODE FOR NEURON N ARE FOUND.
269:          LX          NONEUS
270: *D.1.CHECKS NEURON N JUST FIRED.
271: STCODE   LA,IX       BASE2
272:          SKN
273:          J           DECRX2
274:          SSP                     FIND NEURON JUST FIRED
275:          STA         STPNTR      STORE POINTER
276:          STX         HOLDX
277: *D.2.CHECKS N IS INTEGRATING NEURON.
278:          EX
279:          ALS         1
280:          S           C1
281:          EX
282:          LA,IX       BASE1       IS IT INTEG.?
283:          AND         TESTYP
284:          C           TYPE2
285:          SE
```

```
286:          J          RETEST
287:          ICX        1
288:          LA,IX      BASE1
289:          EQ
290:          LLD        6
291:          CLR
292:          LLD        8
293:          STA        KEEPSB      KEEP INPUT SOB
294:          CLR
295:          LLD        2           LAYER
296: *  THIS IS T REFERED TO IN D.3.3. TESTS FOR 11,T=100
297: *  IF SO,AND STORES IN 'NLAYER'.
298:          C          C3
299:          SNE
300:          A          C1
301:          LLD        12
302:          STA        NLAYER      LAYER IN BITS 1,2,3
303: *D.3.SCANS CONNECTIONS OF N.
304: SKLOOP  CALL       SCAN,STPNTR,RETEST
305: *  RETURNS WITH C(E).
306:          LRS        6
307: *D.3.1.IS N CONNECTED TO CONTEXT NEURON?
308:          ALS        1
309:          S          C1
310:          EX
311:          LA,IX      BASE1       IS IT CONN. TO CONTEXT?
312:          AND        TESTYP
313:          C          TYPE1
314:          SE
315:          J          SKLOOP
316: *D.3.2.CHECKS CODE INCOMPLETE.
317:          ICX        1
318:          LA,IX      BASE1
319:          SKP                    COMPLETED CODE?
320:          J          SKLOOP
321: *D.3.3.IS PART OF CONTEXT CODE ASSOCIATED WITH LAYER
322: *  OF N ALREADY FOUND?
323:          AND        NLAYER      CODE FOR THIS LAYER?
324:          C          C0
325:          SNE
326:          J          STORED
327: *  CODE FOR THIS LAYER FOUND ALREADY. CHOOSES
328: *  RANDOMLY BETWEEN CONNECTIONS.
329:          L          RANDOM      CHOOSE BETWEEN CONNS.
330:          SAE
331:          J          NEWCON
332: *  LEAVES OLD CONNECTION AND PRINTS MARKER.
333:          LA         R0
334:          L          MARKER
335:          J          SKLOOP
336: NEWCON  LA         R1
337: *  CHOOSES NEW CONNECTION AND PRINTS MARKER.
338:          L          MARKER
339: *D.3.4.SETS TAG AND CHECKS CODE HAS ADDRESS.
340: STORED  LA,IX      BASE1
341:          OR         NLAYER      TAG FOR THIS LAYER
342:          STA,IX     BASE1
343:          AND        KPADRS      KEEP ADRESS
344:          C          C0
345:          SE
346:          J          ADRSTR      ADDRESS ALREADY STORED
347: *  CODE NOT ALREADY GIVEN ADDRESS. ADDRESS IS
348: *  CURRENT VALUE OF SOBNUM. THEN UPDATES SOBNUM.
349:          LA,IX      BASE1
350:          A          SOBNUM
351:          STA,IX     BASE1
```

```
352:          AOM       SOBNUM
353: *D.3.5.STORES CODE BY CONSTRUCTING CORRECT
354: *  ADDRESS.                                                    74
355: ADRSTR   LA        NLAYER
356:          LRS       5           1 TO 0,2 TO 1,4 TO 2,
357:          AND       ZEROB8      ZERO BIT 8
358:          STA       CONSAD      CONSTRUCT ADDRESS
359:          LA,IX     BASE1
360:          AND       KFADRS
361:          A         CONSAD
362: *  STORES CODE IN ADDRESS CALCULATED.
363:          EX
364:          LA        KEEPSB
365:          STA,IX    SOBASE      STORED IN BLOCK
366:          J         SKLOOP
367: RETEST   LX        HOLDX
368: DECRX2   DCX       1
369:          J         STCODE      STORE CODE
370:          SSW       8
371:          J         INPUT
372: *  PRINTS SOB CONCENTRATIONS EVERY CYCLE,
373: *  ONCE CODE COMPLETED.
374:          LA        MTWNEU
375:          SSP
376:          EX
377: CHECK    LA,IX     BAS2P1
378:          AND       TESTYP
379:          C         TYPE1
380:          SNE
381:          J         OUTP
382: DCREES   ICX       2
383:          J         CHECK
384:          J         LINE
385: OUTP     ICX       1
386:          LA,IX     BAS2P1
387:          SKN
388:          J         CARIGE
389:          AND       CONCN
390:          LRS       8
391:          CALL      WRITE
392: MISSED   LA        BLANKS
393:          CALL      PRINT
394:          DCX       1
395:          J         DCREES
396: LINE     LA        CRLF
397:          CALL      PRINT
398:          J         INPUT
399: CARIGE   LA        BLANKS
400:          CALL      PRINT
401:          LA        BLANKS
402:          CALL      PRINT
403:          LA        BLANKS
404:          CALL      PRINT
405:          J         MISSED
406:
001: *
002: *
003: ************************
004: *
005: DELTAL   OCT       4           CHANGE AMOUNT OF LEAKBACK
006: DELTAD   OCT       1           CHANGE SPEED OF DIE-AWAY
007: *
008: ************************
009: *
010: *
011: TESTYP   OCT       140000
```

```
012: TYPE0   OCT      0
013: TYPE1   OCT      40000
014: TYPE2   OCT      100000
015: TYPE3   OCT      140000
016: MLAYER  OCT      1
017: HLDSOB  OCT      1774
018: KEEPSB  OCT      0
019: TESTLA  OCT      3
020: ZEROIN  OCT      140377
021: KEEPIN  OCT      37400
022: HOLDX   OCT      0
023: SCNPTR  OCT      0
024: INPSOB  OCT      0
025: HOLDIN  OCT      0
026: C0      OCT      0
027: C1      OCT      1
028: C2      OCT      2
029: C3      OCT      3
030: C17     OCT      17
031: SETBIT  OCT      40000
032: BAS2P1  OCT      0
033: BASE2   OCT      0
034: EMPTY   OCT      0
035: POINTR  OCT      0
036: MNONEU  OCT      0
037: MTWNEU  OCT      0
038: CTEST   OCT      0
039: PULSES  OCT      0
040: TNONEU  OCT      0
041: HOLD    OCT      0
042: NONEUS  OCT      0
043: STORE   OCT      0
044: IN      BCI      1,IN
045: PU      BCI      1,PU
046: T_      BCI      1,T_
047: ME      BCI      1,ME
048: ML      BCI      1,ML
049: 0       BCI      1,0
050: STEP    OCT      0
051: KEEPX   OCT      0
052: STOREA  OCT      0
053: SOBNUM  OCT      0
054: KPADRS  OCT      377
055: ZEROB8  OCT      177577
056: STPNTR  OCT      0
057: NLAYER  OCT      0
058: CONSAD  OCT      0
059: KPTAGS  OCT      170000
060: MLUTAG  OCT      70000
061: CNECTD  OCT      0
062: PBLOCK  OCT      1400
063: CONTX   OCT      0
064: HLDADR  OCT      0
065: C400    OCT      400
066: COMSOB  OCT      170000
067: KPREST  OCT      7777
068: CMPADR  OCT      0
069: COMPRD  OCT      0
070: HELD    OCT      0
071: COMMA   BCI      1,,
072: NFIRED  OCT      0
073: SAVEIN  OCT      0
074: STOREX  OCT      0
075: LIMIT   OCT      6
076: DECAY   OCT      6
077: INTIC   OCT      20000
```

```
078: THRESH  OCT       72
079: FPZERO  OCT       177400
080: BLANKS  BCI       1,
081: NEURNO  OCT       0
082: C11     OCT       11
083: C12     OCT       12
084: C5      OCT       5
085: N5      OCT       -5
086: M12     OCT       -12
087: WRDNUM  OCT       0
088: SHIFT1  OCT       0
089: SHIFT2  OCT       0
090: C20     OCT       20
091: DATBAS  OCT       25000
092: ADDRES  OCT       0
093: SHFTO   OCT       177740
094: DOT     OCT       256
095: COLON   OCT       272
096: SMICLN  OCT       273
097: SPACE   OCT       240
098: C117    OCT       117
099: C1400   OCT       1400
100: START   OCT       0
101: WORDS   OCT       0
102: KEPT    OCT       0
103: KEEPU   OCT       40377
104: ZERSTR  OCT       77700
105: C100    OCT       100
106: C77     OCT       77
107: KEEPCN  OCT       77400
108: LBLOCK  OCT       400
109: ZERBT1  OCT       37777
110: GTRTHN  OCT       276
111: LESTHN  OCT       274
112: RANGE   OCT       177
113: MARKS   OCT       0
114: RANDNO  OCT       0
115: HOLDIG  OCT       0
116: C200    OCT       200
117: R0      OCT       260
118: R1      OCT       261
119: M       OCT       315
120: STAR    OCT       252
121: ZERO    OCT       0
122: C212    OCT       212
123: C215    OCT       215          CR
124: C254    OCT       254
125: C260    OCT       260
126: C267    OCT       267
127: NUM     OCT       0
128: CEEP    OCT       0
129: DFTAPE  OCT       100
130: DE      BCI       1,DE
131: CRLF    OCT       106612
132: N6      OCT       -6
133: COUNT   OCT       0
134: TENCYS  OCT       0
135: NOMARK  OCT       0
136: M120    OCT       -120
137: CONCN   OCT       77400
138: *
139: **********************
140: *
141: *
142: ***********
143: **  SUBROUTINES FOLLOW.
```

```
144: **********
145: *
146: *
147: *****
148: *
149: * READS DATA FROM PAPER TAPE
150: *
151: *****
152: *
153: READ1    ADR         0
154:          LA          N6
155:          STA         COUNT
156:          CLR
157:          STA         NUM
158: RLOOP    SI          4           STOPPED?
159:          LRS         6
160:          SAE
161:          J           RLOOP
162:          SI          4           READY?
163:          SAE
164:          SKU
165:          J           *-3
166:          LA          DFTAPE
167:          DF          4           START
168:          DI          4
169:          STA         CEEP
170:          CLR
171:          DF          4           STOP
172:          LA          CEEP
173:          C           ZERO        DISREGARD LEADER
174:          SNE
175:          J           RLOOP
176:          C           C212        DISREGARD LINE FEED
177:          SNE
178:          J           RLOOP
179:          C           C254        COMMA EXIT
180:          SNE
181:          J           REXIT
182:          C           C215        CR EXIT
183:          SNE
184:          J           REXIT
185:          C           C260        DIGIT CHECK
186:          SGE
187:          J           DATER       DATA ERROR
188:          C           C267
189:          SLE
190:          J           DATER
191:          LRD         3
192:          LA          NUM
193:          LLD         3
194:          STA         NUM
195:          AOM         COUNT
196:          J           RLOOP
197: DATER    LA          DE             DATA ERROR
198:          CALL        PRINT
199:          LA          CRLF
200:          CALL        PRINT
201:          J           READ1+1
202: REXIT    EQ                         STORE COMMA OR CR
203:          LA          NUM
204:          J,I         READ1
205: *
206: *
207: *****
208: *
209: * PRINTS MARKERS OR NOT
```

```
*
*****
*
MARKER  ADR     0
        SSW     E
        J,I     MARKER
        CALL    PRINT
        AOM     NOMARK      NO. OF MARKERS
        J,I     MARKER
        LA      M120
        STA     NOMARK
        LA      CRLF
        CALL    PRINT
        J,I     MARKER
*
*
*****
*
* MARKS OFF PAGE FOR OUTPUT
*
*****
*
DOTTED  ADR     0
        LA      BLANKS
        CALL    PRINT
        LA      BLANKS
        CALL    PRINT
        LA      BLANKS
        CALL    PRINT
        LX      C117
        LA      C11
        STA     MARKS
MKLOOP  LA      MARKS
        C       C0
        SE
        J       DOTS
        LA      C11
        STA     MARKS
        LA      C254
        CALL    PRINT
DECRX5  DCX     1
        J       MKLOOP      MARK LOOP
        J,I     DOTTED
DOTS    S       C1
        STA     MARKS
        LA      DOT
        CALL    PRINT
        J       DECRX5
*
*
*****
*
* GENERATES RANDOM NUMBER
*
*****
*
RANDOM  ADR     0
        LA      RANDNO
        C       C0
        SNE
        LA      RANDOM      SUBSTITUTE IF 0
        STA     RANDNO
        AND     C1
        STA     HOLDIG
        LA      RANDNO
        AND     C200
```

```
276:          LRS         7
277:          C           HOLDIG
278:          LA          RANDNO
279:          SE
280:          SSN
281:          LRS         1
282:          STA         RANDNO
283:          J,I         RANDOM
284: *
285: *
286: *
287:          END         0
288: @
289:
289: _
```

1.2 Listing of Source File LAST

```
001: *
002: *
003: *
004: *         ********
005: *         *      *
006: *         * LAST *
007: *         *      *
008: *         ********
009: *
010: *
011: *   THIS PROGRAM LOADED LAST
012: *
013: *
014:           REL       0
015:           NAME      BASE1,SOBASE
016: *
017: *
018: *   SETS SIZE OF BLOCK 1 AT 1023 CELLS AND
019: *   ENSURES THAT BASE1 FOLLOWS ON DIRECTLY
020: *   AFTER PROGRAM.
021: *
022: *
023: SOBASE  ADR       *
024:           BSS       1023,0
025: BASE1    ADR       *
026: *
027: *
028:           END       0
029: @
030:
030: _
```

Appendix 2. Preliminary Study for a Tactile Mobility Aid for the Blind

2.1 INTRODUCTION

The work described here is the result of a preliminary study to determine the feasibility of developing a mobility aid for the blind which would use an ultrasonic scanning array to produce tactile images on an abdomen-mounted matrix.

The fact that the device is to be designed for mobility (rather than, say, reading) is important, since mobility is a specialised task. For example:

(i) the consequences of making a mistake when negotiating an environment can be much worse than when reading

(ii) mobility is public, while reading is a private affair. Hence some grace is needed.

(iii) while it is possible to read at one-tenth the speed of a normal sighted person, walking at one-tenth the normal speed causes difficulties due to the loss of 'gait'.

Reasons for considering a tactile blind-aid include the following:

(i) the aid would be useful for deaf-blind people

(ii) the skin (especially the abdomen) is not normally being used for other (i.e. sensory) purposes

while the advantages of ultrasonics include the following:

(i) as a stress wave, it interacts mechanically with the environment (i.e. glass appears as a barrier)

(ii) its relatively long wavelength makes the environment 'shiny' and hence it performs some preprocessing.

(iii) it is easier to produce a plan view of the environment than it would be using an optical system.

The reasons for considering a plan view rather than an image view include the following:

(i) it seems that a lower resolution would be sufficient

(ii) the centre flow point of a visual system has the lowest resolution, which is probably the worst place for it. A plan view flows from one edge when there is movement.

(iii) image systems in the past have not met with much success for mobility (Bach-y-Rita, 1972)

(iv) the processing would probably be simpler.

The following section summarises the various ultrasonic devices considered for the sensory aid. Factors relevant to the choice of one of these include:

(i) resolution, in range and bearing

(ii) whether multiple targets can be detected in the same range resolution annulus or not

(iii) physical size of array, considering the effects of number of array elements and aperture distribution on sidelobe structure and power transmitted

(iv) power consumption, since portability is important

(v) cosmetic acceptability (also related to size)

(vi) reliability

(vii) complexity and cost.

2.2 THE DEVICES CONSIDERED

1. An Impulse-Scanning System (Kay, 1961)

The receiver is shown in Figure 15(a). In principle the system consists of an array of transducer elements, each one successively tapped to a single delay line. Signals received by the transducers are translated to steadily increasing frequencies by modulation with a linearly changing frequency. Since when a signal is so modulated its phase-angle is unaltered, and since a delay-line phase-shift increases linearly with frequency, the effect of the modulation is to steer the received beam steadily across the scanning sector.

If the modulation frequency is swept through its cycle twice during each pulse duration, all directions of the sector are sampled without loss of information. The output of the delay line at a particular instant is an addition of all the modulated inputs, each one delayed in phase by an amount dependent upon the value of the modulating frequency at that time and its position on the delay line. Outputs which are in phase correspond to a target detected at the angle at which the receiving beam is 'pointing'. Hence multiple targets in the same range annulus can be detected.

The major problem with the system appears to be the analogue delay line, which would be difficult to construct in a compact form. This system is compared with a possible frquency-scanned system in section 2.3.

2. The Use of Surface Acoustic Wave (SAW) Devices

SAW devices may be a solution to the problem of obtaining the analogue time delays required above. Whitehouse (1974) considers the design of scanning arrays using these devices. Advantages noted in using SAW devices in such applications were reliability, ease of fabrication, small size, and low cost.

However, very little literature exists concerning fabrication techniques. A method of photo-etching is used, in a manner similar to integrated circuits, on a piezoelectric quartz substrate. It seems that such devices are not yet available commercially.

3. The Acoustic Lens

It seems possible that by directing the amplified output of a transducer array to the pupil of an acoustic lens by piezoelectric transducers, the analogue of an optical focusing system could be obtained. A real-time Fourier transform of the spatial frequencies in the environment would be performed at the focus of the lens. Problems that would be involved include:

(i) large reflections from the walls of the lens

(ii) mismatch between transducers and the lens

(iii) aberration.

A possible lens that could reduce the problem of reflections is shown in Figure 15(b). For focusing to take place within a reasonable length the lens would need to be solid or liquid. In the diagram, holes somewhat smaller than the wavelength of propagation in the aluminium block are drilled at either face.

By increasing the 'density' of the holes toward each face, and imbedding each face in absorbent rubber so that the holes become filled with rubber, a gradual match could be obtained between the aluminium and the rubber. The system would be simple, reliable, and operate in real time.

Some references appear in the literature to solid and liquid lenses in array systems (Toulis, 1963; Boyles, 1965).

4. The Interferometer

The traditional interferometric system using two receiving transducers was considered. While such a system is relatively simple compared to an array, there are two major disadvantages:

(i) the small phase differences detected at each transducer from a particular target mean that a relatively large transducer spacing is required. This would not be cosmetically acceptable in a head-mounted device.

(ii) ambiguities result when there is more than one target per range annulus.

5. The Digital Sonar

A system constructed by Danh (1975) uses an array of seven transducers. Instead of phasing the array in the normal manner to steer the beam, the phase differences between adjacent points on the array are measured. These measurements are used to compute the direction of arrival of the wavefront. The average of the measurements gives the bearing of the target. The system is essentially

digital in that after amplification the transducer outputs are gated to quantise the phase differences. Some work on this sytem and subsequent conclusions are described in section 2.4.

6. A Frequency-Scanned System

The transmitter for this system is shown in concept in Figure 15(c). For n = 0, a square wave of frequency

$$\frac{10\ \text{MHz}}{64} = 156\ \text{KHz}$$ appears at the first transducer.

It takes exactly one period to be clocked through the shift-register to the second transducer, and hence the outputs are in phase and the transmitted beam points broadside.

For n non-zero, the phase of the second transducer relative to the first is

$$\phi = 2\pi \left[1 - \left(\frac{64}{64+n}\right)\right] = \frac{2\pi n}{64+n}$$

Also, the radian frequency at the input to the transducers is:

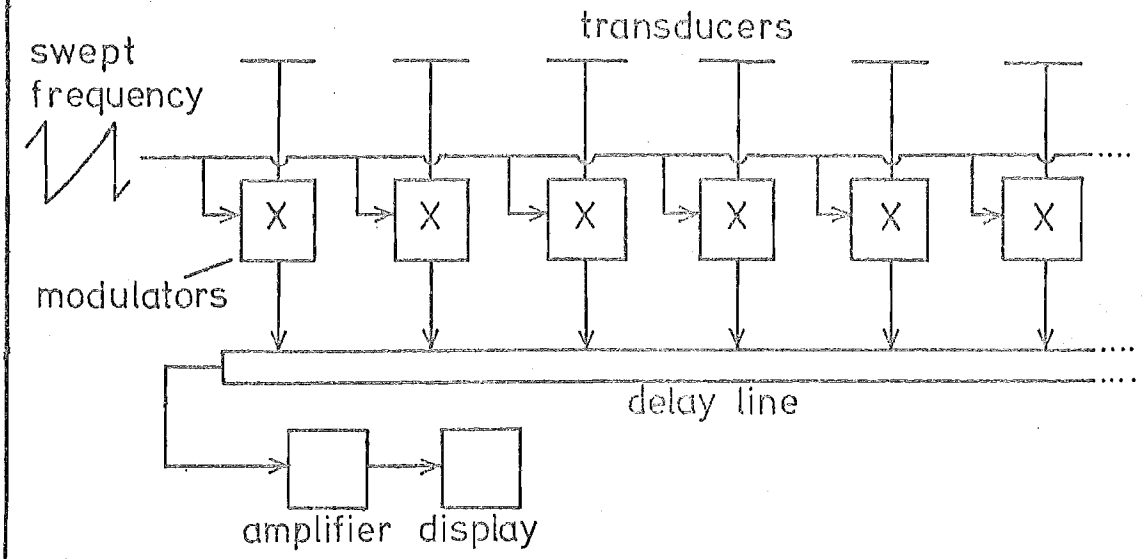
$$\omega = \frac{2\pi f}{64+n}$$

where f is the clock frequency. So the time delay between transducers is

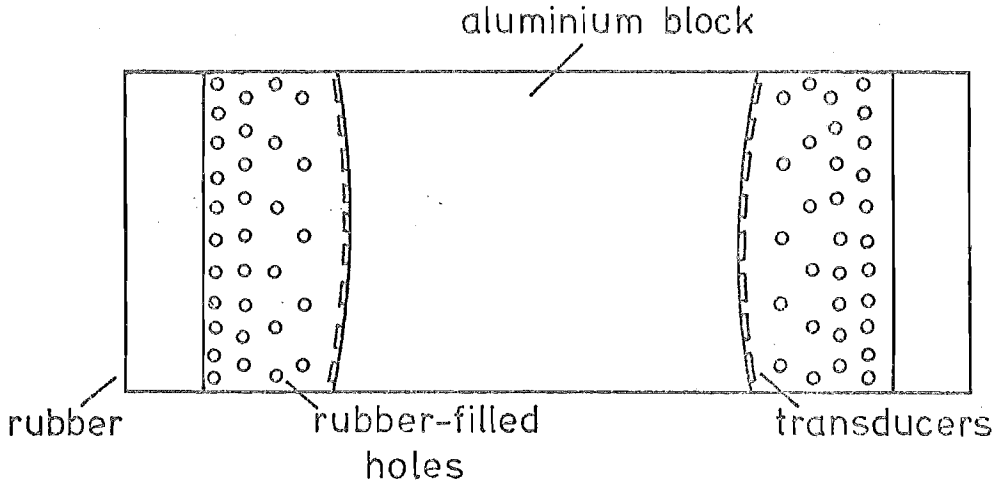
$$t = \frac{\phi}{\omega} = \frac{n}{f}$$

Hence the time delay between transducers is proportional to the value of n, and so beam direction is also.

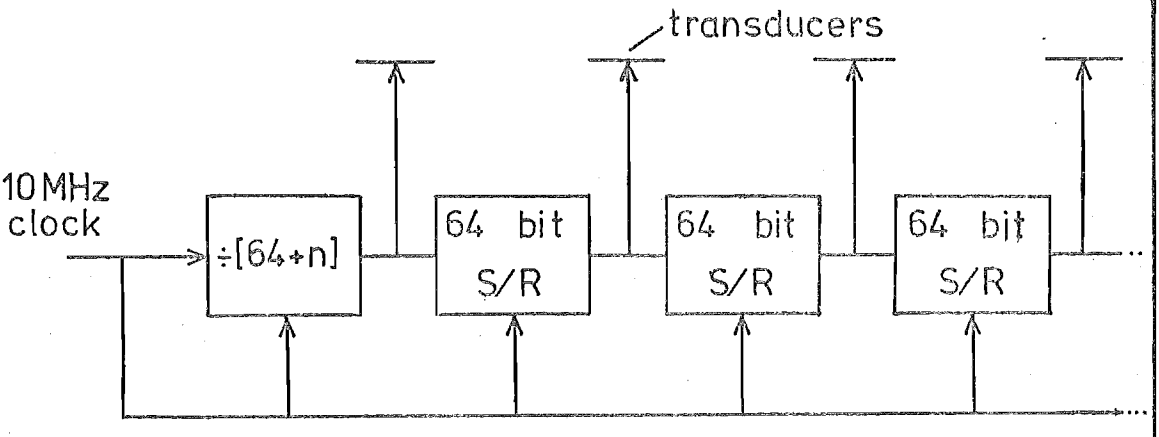
The above suggests that the beam can be scanned (periodically or aperiodically) across a sector.



(a) impulse-scanning receiver



(b) acoustical lens



(c) frequency-scanning transmitter

FIGURE 15. APPENDIX 2.

Range information can be obtained by starting a timer when transmission begins in a particular direction. Signals are received by a single wide-band receiver followed by a bank of filters, one for each transmission direction. Response from a particular filter means that there is a target in its particular bearing-cell.

This system can be considered as the inverse of the impulse-scanning system described earlier. The two systems are compared in the next section.

2.3 COMPARISON OF IMPULSE- AND FREQUENCY-SCANNING SYSTEMS

It is useful to compare the signal-to-noise ratios (SNR's) of the two systems. Following the argument of Lindars (Tucker, 1960) the SNR of an impulse-scanning system is not influenced by the scanning rate and is the same as when the beam points at the target continuously. This is because although the received pulse would be n times as long as it would be if the entire sector was being scanned (n is the number of beam-widths in the sector), the amount of noise received at this angle would also increase n times. If this situation is now compared with a conventional echo-ranging system, the only difference is that transmission is occurring in n sectors rather than in one. So the received pulse is $\frac{1}{n}$ times the intensity of the conventional case and the SNR is reduced by $\frac{1}{n}$.

If the frequency-scanning system is considered as a series of conventional echo-ranging transmissions, then the only difference between one such transmission and the normal

echo-ranging system is that reception is taking place over n-1 sectors rather than over one sector (since one receiver channel will always be blanked if it is assumed that transmission is always occurring in one sector).
If transmission is occurring in only one sector the SNR is decreased by 1/(n-1) compared with conventional echo-ranging. Hence the SNR's of the two systems are only marginally different.

The delay line of the impulse-scanning system is also similar to the filters of the frequency-scanning system, as both require a phase-shift proportional to frequency.

The frequency-scanning system seems to require less hardware than the impulse-scanning system, although neither could be built cheaply in a compact form at present.

2.4 THE DIGITAL AIR SONAR: CONCLUSIONS

Some time was spent attempting to complete the Air Sonar model, constructed by Danh (1975). Although the major part of the digital circuitry was working (although unreliably) the problem of pre-amplifier oscillation had not been solved. This problem was finally found to be due to connections between pins of unused turns of the pre-amplifier auto-transformer and unused sections of VERO-board.

Work was stopped on the Scanning Air Sonar for the following reasons:

(i) reliability; the complete unit has never been in operation at one time. LSI techniques would improve this.

(ii) the system can detect only one target per range resolution annulus

(iii) the system performs a great deal of decision-making before the display. This is not necessarily good in a mobility aid.

(iv) a new transmitter, locked to the receivers, needs to be built for bearing information to be displayed accurately.

(v) the beam width is narrow (less than 30^{o}). A reduction in operating frequency would widen the beam, but then transducer sensitivity would drop markedly. Also the transmitter oscillator third-harmonic would move further into the transducer pass-band to distort the received waveform (accurate operation depends on very little distortion).

2.5 DESIGN PHILOSOPHIES ENCOUNTERED

1. The blind-aid system should be designed from the point of view of the requirements of the user rather than from what we are capable of making. This approach means that receiver specifications must be matched to the most information that the user can resolve with the tactile display. For an array of transducers, the resolution at the display is related to the receiver beamwidth which, in turn, depends on the number of array elements used.

Hence the resolution that can be attained with the tactile display must be determined to decide how many elements the transducer array should have.

The problem with this approach is the determination of tactile resolution. The traditional two-point tests appear to be of no use. Instead, resolution has been found to depend on:

(i) the temporal nature (or otherwise) of patterns (Davis, 1968; Eskildsen *et al.*, 1969)

(ii) the type of stimulator (Vierck and Jones, 1969)

(iii) learning and recognition (Bach-y-Rita, 1972)

(iv) time of the day

(v) temperature

(vi) social pressures.

2. The conclusion of Bach-y-Rita (1972) is 'to present as much of the raw information as possible to enable the brain to select the appropriate information for each task'. The result of this approach is that the user is not considered, and design ultimately becomes restricted only by cost.

3. A third approach encountered is to design the sensory interface so that it transmits no more than the least information possible for the subject to have mobility. This method is likely to be less costly than that just described.

However, the possibility exists that with sufficient training the subject may finally become limited by the ability of the device to transmit information rather than his ability to assimilate the information it presents.

2.6 REFERENCES CONSULTED FOR APPENDIX 2

Anderson V.C. 'Digital Array Phasing', JASA, 32, 1960, p.867

Bach-y-Rita P. 'Brain Mechanisms in Sensory Substitution', Academic Press, 1972

Born N., Lances C.J., Honkorp J., Hugenholtz P.G. 'Ultrasonic Viewer for Cross-Sectional Analysis of Moving Cardiac Structures', Bio-Medical Eng. 6, Nov. 1971, p.500

Boyles T. 'Theory of Focusing Plane Waves by Spherical Liquid Lenses', JASA 38, 1965, p.393

Cook G.B. 'The Ultrasonic Camera: A Study of the Transducer-Computer Interface', M.E. thesis, Univ. of Cant., 1969

Danh B.C. 'A Scanning Air Sonar', M.E. thesis, Univ. of Cant., 1975

Davis H. 'Epilogue: A Chairman's Comments on the Neural Organization of Sensory Systems', *In* 'The Skin Senses' ed. Kenshalo D.R. (q.v.), p.589

Dews P.B., Wiesel T.N. 'Consequences of Monocular Deprivation on Visual Behaviour in Kittens', J. Physiol. 206, 1970, p.437

Eskildsen P., Morris A., Collins C.C., Bach-y-Rita P. 'Simultaneous and Successive Cutaneous Two-Point Thresholds for Vibration', Psychon. Sci. 14, 1969, p.146

Folds D.L., Brown D.H. 'Focusing Properties of Cylindrical Liquid-Filled Acoustic Lenses with Large Diameter-to-Wavelength Ratios', JASA 43, n.3, 1968, p.560

Harrold S.O., West R.C. 'Far-Field Sector Scanning Using a Sampled Multielement Array', Elect. Lett. 7, n.4, Feb. 1971

Heighway J. 'Surface Acoustic Wave Devices and Appliances', Systems Tech. 23, March 1976

Hollard M.G., Clairborne L.T. 'Practical Surface Acoustic Wave Devices', Proc. I.E.E.E. 62, 1974, p.582

Kanerskii I.N., Surikov B.S. 'Homogeneous Spherical and Cylindrical Lenses for the Focusing of Sound Waves', Sov. Phys.-Acoust. 17, 1971, p.43

Kay L. 'A Comparison Between Pulse and Frequency-Modulation Echo-Ranging Systems', J. Brit. I.R.E. 19, n.2, Feb. 1959, p.105

Kay L. 'Progress in Underwater Echo-Ranging', Brit. Comm & Elect., Oct. 1961

Kay L. 'Ultrasonic Image Synthesis', *In* 'Research Techniques in Nondestructive Testing', Vol. 2, ed. Sharpe R.S., Academic Press 1973, Chapter 12

Kay L. 'The Design and Evaluation of a Sensory Aid to Enhance Spatial Perception of the Blind', Dept of Elect. Eng. Report, Univ. of Cant., Jan. 1973, No.21

Kay L. 'A Sonar Aid to Enhance Spatial Perception of the Blind: Engineering Design and Evaluation', Radio and Electronic Engineer 44, n.11, Nov. 1974, p.605

Kay L. 'Orientation for Blind Persons: Clear Path Indicator or Environment Sensor', New Outlook for the Blind, Sept. 1974, p.289

Kenshalo D.R. (ed.) 'The Skin Senses - Proceedings of the First International Symposium on the Skin Senses Held at the Florida State University in Tallahasse', Thomas, Illinois, 1968

Lewin R. 'Seeing With a Blind Eye', New Scientist, June 1975, p.696

Mackworth N.H., Morandi A.J. 'The Gaze Selects Informative Details Within Pictures', Percept. Psychophys. 2, 1967, p.547

Miller G.A. 'The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information', Psychol. Rev. 63, 1956, p.81

Morgan D.P. 'Surface Acoustic Wave Devices - 1. An Introductory Review', Ultrasonics 7, 1973, p.121

Nye P.W., Bliss J.C. 'Sensory Aids for the Blind: A Challenging Problem with Lessons for the Future', Proc. I.E.E.E. 58, 1970, p.1878

Sangster F.L.J. 'The Bucket-Brigade Delay Line, a Shift Register for Analogue Signals', Philips Tech. Rev. 31, n.4, 1970, p.92

Skolnik M.I. 'Introduction to Radar Systems', McGraw-Hill, 1962

Somer J.C. 'Electronic Sector Scanner for Ultrasonic Diagnosis', Ultrasonics 6, July 1968, p.153

Sterling T.D., Bering E.A., Pollack S.V., Vaughan H.G. (eds) 'Visual Prosthesis - The Interdisciplinary Dialogue. Proceedings of the Second Conference on Visual Prosthesis', Academic Press, 1971

Toulis W.J. 'Acoustic Focusing with Spherical Structures', JASA 35, 1963, p.286

Tucker D.G. 'Directional Echo-Sounding. Some Possible Improvements in Equipment and Technique', Int. Hydro. Rev. 37, n.2, July 1960

Tucker D.G. 'Some Possibilities in Civil Underwater Echo-ranging', J. Brit. I.R.E. 20, n.4, 1960

Tucker D.G. 'Electronic Sector-Scanning ASDIC', J. Inst. Navigation, 12, n.2, April 1959

Tucker D.G., Welsby V.G., Kendell R. 'Electronic Sector Scanning', J. Brit. I.R.E. 18, n.8, Aug. 1958, p.465

Vierck C.J., Jones M.B. 'Size Discrimination on the Skin', Science, 1963, 1969, p.488

Voglis G.M., Cook J.C. 'Underwater Applications of an Advanced Acoustic Scanning Equipment', Ultrasonics 7, Jan. 1966

Welsby V.G. 'Electronic Sector-Scanning Array using a Multi-Frequency Carrier', Elect. Tech., Jan. 1962, p.13

Welsby V.G., Tucker D.G. 'Multiplicative Receiving Arrays', J. Brit. I.R.E. 19, n.6, June 1959, p.369

Whitehouse H.J. (ed.) 'Optical and Acoustical Signal Processing', Symp. Opt. & Acoust. Micro-Elect. April 16, 174, Polytechnic Press, N.Y., 1974

Whittingham J.A. 'A Hand-Held Electronically-Switched Array for Rapid Ultrasonic Scanning', Ultrasonics 14, n.1, Jan. 1976